



UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE

Corso di Laurea Triennale in Informatica

Algoritmi per l'Approssimazione di Sotto-Isomorfismi tra Impronte Digitali

*Algorithms for the Approximation of Sub-Isomorphisms
between Fingerprints*

CANDIDATO:

Marco Corazzini

RELATORE:

Prof. Alessandro Dal Palù

MATRICOLA:

332177

*“Coloro che possono immaginare qualsiasi cosa,
possono creare l'impossibile.”*

Alan M. Turing

A mia nonna Antonietta

Indice

Introduzione	1
1 Impronte digitali per l'analisi forense	3
1.1 Struttura delle impronte digitali	3
1.2 Classificazioni	4
1.2.1 Classificazione Galton-Henry	4
1.2.2 Classificazione Gasti	6
1.2.3 Minuzie	8
1.3 Digitalizzazione delle impronte digitali	9
1.3.1 Grafi e Isomorfismi	10
1.4 Metodi di matching	13
1.5 AFIS	13
1.5.1 SourceAFIS	14
1.5.2 Funzionamento dell'algoritmo	15
1.6 Ereditarietà delle impronte digitali	18
2 Dataset	19
2.1 Creazione del dataset	19
2.2 Catalogazione delle impronte	20
2.3 Digitalizzazione delle impronte	21
2.4 Dimensione del dataset	21
3 Algoritmi per la ricerca di sotto-isomorfismi	23
3.1 Preprocessing	23
3.2 Ricerca tramite sotto-grafi	25
3.2.1 DFS	26
3.2.2 BFS	28
3.2.3 Controllo dell'isomorfismo	31
3.3 Ricerca tramite nodi	36
3.3.1 Analisi del singolo nodo	36
3.3.2 Analisi di nodi in una matrice	38

4 Risultati	43
4.1 Metodi di Scoring	43
4.1.1 Scoring razionale	44
4.1.2 Scoring empirico	45
4.2 Analisi dei test	46
4.2.1 Risultati algoritmo DFS	46
4.2.2 Risultati algoritmo BFS	47
4.2.3 Risultati algoritmo Singoli Nodi	49
4.2.4 Risultati algoritmo Matrice	50
4.2.5 Considerazioni	51
Conclusione	53
Bibliografia	55

Elenco delle figure

1.1	Dermatoglifi	4
1.2	Esempi di Arch	5
1.3	Esempi di Loop	6
1.4	Whorl	6
1.5	Tipologie di impronte del metodo Gasti	7
1.6	Lista di minuzie	8
1.7	Esempio di grafo non orientato	10
1.8	Esempio di isomorfismo tra grafi	11
1.9	Esempio di Biezione	11
1.10	Esempio di sotto-isomorfismo tra grafi	12
1.11	Estrazione delle minuzie di SourceAFIS	15
1.12	I colori sono determinati in base alla lunghezza e all'angolo degli archi. Archi simili hanno colori simili.	16
1.13	Il nodo minuzia è blu, L'albero di pairing è verde mentre il grafo che supporta i vertici è giallo.	17
3.1	Impronta 01_Ay_01_00 prima e dopo il preprocessing, gli archi eliminati sono di colore nero	24
3.2	Esempio di albero DFS. L'output è il seguente : 0, 1, 4, 5, 2, 6, 3, 7	26
3.3	Impronta 01_Ay_01_00 dopo aver eseguito l'algoritmo DFS . . .	27
3.4	Box contenente un sotto-grafo trovato tramite l'algoritmo DFS .	28
3.5	Esempio di albero BFS. L'output è il seguente : 0, 1, 2, 3, 4, 5, 6, 7	29
3.6	Impronta 01_Ay_01_00 dopo aver eseguito l'algoritmo BFS . . .	30
3.7	Box contenente un sotto-grafo trovato tramite l'algoritmo BFS .	31
3.8	Esempio di isomorfismo trovato tramite DFS	34
3.9	Esempio di isomorfismo trovato tramite BFS	35
3.10	Mappatura di ogni nodo dopo la fase di prepocessing	37
3.11	Riscontro dei nodi di match	37
3.12	Mappatura degli indici della matrice	40

3.13 Rappresentazione della matrice sull'impronta 01_Ay_01_00. Le celle rosse rappresentano celle con nodi al proprio interno, mentre le celle nere rappresentano celle totalmente vuote.	41
4.1 Grafico della funzione di scoring	45

Elenco degli algoritmi

1	Preprocessing dei bordi	24
2	Esempio di implementazione della box tramite il suo range	25
3	Depth-First Search (DFS)	26
4	Breadth-First Search (BFS)	29
5	Creazione albero DFS	33
6	Isomorfismo tra alberi tramite ordinamento di stringHash	33
7	Match dei singoli nodi	37
8	Definizione della dimensione massima della matrice	39
9	Creazione delle celle della matrice	39
10	Ricerca indice e inserimento	40

Elenco delle tabelle

4.1	DFS senza preprocessing cardinalità = 3, box creazione = 15, box comparazione = 30	46
4.2	DFS con preprocessing cardinalità = 3, box creazione = 15, box comparazione = 30	47
4.3	DFS senza preprocessing cardinalità = 5, box creazione = 15, box comparazione = 30	47
4.4	DFS con preprocessing cardinalità = 5, box creazione = 15, box comparazione = 30	47
4.5	BFS senza preprocessing cardinalità = 3, box creazione = 15, box comparazione = 30	48
4.6	BFS con preprocessing cardinalità = 3, box creazione = 15, box comparazione = 30	48
4.7	BFS senza preprocessing cardinalità = 5, box creazione = 15, box comparazione = 30	48
4.8	BFS con preprocessing cardinalità = 5, box creazione = 15, box comparazione = 30	48
4.9	Singoli Nodi con box comparazione = 1	49
4.10	Singoli Nodi con box comparazione = 2	49
4.11	Singoli Nodi con box comparazione = 4	49
4.12	Singoli Nodi con box comparazione = 8	50
4.13	Score razionale, matrice 15×15	50
4.14	Score razionale, matrice 30×30	50
4.15	Score empirico, matrice 15×15	51
4.16	Score empirico, matrice 30×30	51

Introduzione

Le impronte digitali rappresentano da decenni uno degli elementi fondamentali nell'ambito dell'analisi forense, giocando un ruolo cruciale nelle indagini sulle scene del crimine. La loro unicità e permanenza nel tempo le rendono uno strumento insostituibile per l'identificazione individuale. Ogni impronta digitale possiede un modello distintivo di creste e solchi, che è unico per ciascun individuo e rimane invariato per tutta la vita. Questa caratteristica ha portato allo sviluppo di numerosi metodi e tecniche per il loro rilevamento e analisi.

Nel corso degli anni, la ricerca scientifica e tecnologica ha cercato di migliorare le metodologie per confrontare impronte digitali sconosciute con quelle già note conservate nei database forensi. Tradizionalmente, l'analisi di tali impronte è stata basata su comparazioni visive e meccaniche, ma l'evoluzione della tecnologia ha introdotto strumenti avanzati basati su algoritmi e intelligenza artificiale. Recenti studi hanno dimostrato che le impronte digitali non solo sono uniche tra individui non correlati, ma presentano anche somiglianze significative tra membri di gruppi familiari.

Questa tesi di ricerca si propone di sviluppare e perfezionare algoritmi innovativi volti a identificare le affinità tra le impronte digitali di individui campione e diversi gruppi familiari. L'obiettivo principale è creare strumenti che possano determinare con maggiore precisione il grado di somiglianza tra un'impronta digitale sconosciuta e le impronte di gruppi familiari noti, facilitando così l'identificazione e l'investigazione in ambito forense.

Nel primo capitolo, verrà fornita una panoramica approfondita della storia delle impronte digitali, con una particolare attenzione alle loro caratteristiche uniche e alle metodologie di classificazione sviluppate nel tempo. Saranno esaminati i principali metodi di rilevamento e confronto delle impronte, nonché le tecnologie emergenti che hanno rivoluzionato questo campo. Inoltre, verrà presentato un algoritmo di esempio attualmente in fase di sviluppo, illustrando le sue capacità e i suoi limiti nel contesto del matching delle impronte digitali.

Il secondo capitolo si focalizzerà sull'analisi dettagliata del dataset utilizzato per questa ricerca. Sarà descritta la composizione del dataset, la sua provenienza e la metodologia di suddivisione in categorie pertinenti per l'analisi.

Il terzo capitolo esplorerà il lavoro pratico svolto durante il periodo di ricerca, con una dettagliata esposizione degli algoritmi ideati e sviluppati. Ogni algoritmo sarà descritto nei minimi dettagli, includendo la teoria alla base e la struttura implementativa. Saranno analizzati i risultati preliminari ottenuti e le sfide affrontate durante il processo di sviluppo.

Infine, il quarto capitolo presenterà i risultati finali dei test condotti sugli algoritmi sviluppati, offrendo una valutazione critica delle loro prestazioni. Saranno discussi i punti di forza e le aree di miglioramento individuate, concludendo con una riflessione sulle implicazioni dei risultati per il campo dell'analisi forense e suggerendo possibili direzioni per future ricerche.

Capitolo 1

Impronte digitali per l'analisi forense

L'interesse umano per le impronte digitali risale a migliaia di anni fa, con testimonianze che vanno dall'arte preistorica all'uso di impronte come sigilli nei contratti nell'antica Babilonia. Nel corso dei secoli, diverse culture hanno riconosciuto l'unicità delle impronte digitali, come la Cina la quale le utilizzava già nel XIII secolo per l'identificazione criminale. Nel XIX secolo, studi in Europa e India consolidarono l'idea delle impronte come metodo affidabile per l'identificazione, portando allo sviluppo di sistemi di classificazione e al loro impiego in indagini forensi.[3]

1.1 Struttura delle impronte digitali

Le impronte digitali sono tracce lasciate dai polpastrelli su superfici lisce, originate dalla conformazione delle creste e dei solchi presenti sulla pelle. Questi disegni, noti come *dermatoglifi* [Figura 1.1], sono unici e inalterabili poiché derivano dagli strati profondi del derma. Si sviluppano durante l'ottavo mese di gestazione e rimangono immutati per tutta la vita. La loro individualità è tale che la probabilità di trovare due impronte digitali perfettamente identiche è considerata praticamente impossibile, data l'enorme varietà di combinazioni possibili.[4]

La premessa sull'individualità delle impronte digitali è stata confermata attraverso uno studio matematico condotto nei primi anni del Novecento. L'obiettivo dello studio era determinare il massimo numero di coincidenze possibili tra le impronte delle dieci dita delle mani, considerando la stima della popolazione mondiale dell'epoca, circa 1,65 miliardi di persone. Lo studio prese in esame 16 miliardi di possibili tipologie di impronte e dimostrò che per ottenere almeno 17 coincidenze sarebbero state necessarie più di 17 miliardi

di impronte, un numero superiore all'intera popolazione mondiale del tempo. Questo esperimento confermò quindi l'impossibilità di una totale coincidenza delle impronte tra individui diversi. Grazie alle loro caratteristiche di *immutabilità e unicità*, le impronte digitali sono da tempo utilizzate come metodo di identificazione personale.



Figura 1.1: Dermatoglifi

1.2 Classificazioni

Una volta stabilita l'unicità di ciascuna impronta digitale, è diventato necessario sviluppare un metodo pratico per la sua analisi. Con il passare del tempo, si è scoperto come ogni impronta presenti caratteristiche ricorrenti le quali possono essere osservate su diverse dita. Tali caratteristiche sono state successivamente oggetto di studio e classificazione secondo vari sistemi e ideologie.

1.2.1 Classificazione Galton-Henry

In Inghilterra e in Galles, l'uso delle impronte digitali per l'identificazione dei criminali iniziò nel 1901. Questo sistema si basava sulle teorie di *Francis Galton*, che furono successivamente adattate e perfezionate da *Edward Richard Henry*, allora commissario capo della polizia metropolitana di Londra. La classificazione *Galton-Henry* fu pubblicata nel giugno del 1900 e adottata rapidamente da vari uffici di polizia in diverse giurisdizioni. Oggi, questo sistema rimane in uso in tutti i Paesi anglofoni.

Henry descrisse tre categorie per etichettare un impronta:

- Arch
- Loop
- Whorl

Arch

In questa prima classe le *creste* assumono una forma simile a quella di un *arco* [Figura 1.2]. Si osserva come avvenga una sottodivisione ulteriore in base al tipo di arco trovato:

- *Plain Arch*, un semplice arco in cui le creste partono da un lato, crescono verso il centro dell'impronta per poi scendere e uscire dall'altro lato.
- *Tented Arch*, simile al Plain Arch ma con la presenza di un angolo accentuato al centro con la presenza intrinseca di un delta.



(a) Plain Arch



(b) Tented Arch

Figura 1.2: Esempi di Arch

Loop

La seconda classe descritta da Henry rappresenta un impronta in cui le *creste* entrano da un lato e successivamente si ripiegano su se stesse per poi uscire dallo stesso lato di ingresso [Figura 1.3]. Anche in questo caso possiamo osservare due tipologie di *Loop*:

- *Left Loop*, con ingresso e uscita dal lato *sinistro*
- *Right Loop*, con ingresso e uscita dal lato *destro*



(a) Left Loop



(b) Right Loop

Figura 1.3: Esempi di Loop

Whorl

Come terza e ultima classe, le *creste* compiono un percorso completo attorno al centro dell'impronta, creando una forma ovale lungo tutta l'impronta [Figura 1.4].



Figura 1.4: Whorl

1.2.2 Classificazione Gasti

Nel 1902, *Francesco Leonardi*, allora vicario della polizia giudiziaria italiana, decise di istituire una scuola per la polizia scientifica e nominò direttore il dottor *Salvatore Ottolenghi*, allievo di *Cesare Lombroso*, pioniere della criminologia scientifica. *Ottolenghi* scelse come collaboratore *Giovanni Gasti*,

CAPITOLO 1. IMPRONTE DIGITALI PER L'ANALISI FORENSE

affidandogli il compito di selezionare e addestrare i futuri cadetti della polizia scientifica.

Durante il suo servizio presso la scuola di *segnalamento e identificazione*, Gasti adattò e modificò il sistema di classificazione di *Francis Galton ed Edward Henry*, sviluppando un metodo innovativo per l'identificazione delle impronte digitali. Questo sistema, conosciuto come *identificazione decadattiloscopica* o *metodo Gasti*, divenne una pietra miliare nella polizia scientifica [7].

Il metodo Gasti si basa su tre differenti tipologie di linee:

- *centrali*: passanti per il centro dell'impronta
- *marginali*: passanti per i lati dell'impronta
- *basali*: passanti parallele alle piegature del dito

Grazie a queste tre tipologie di linee, il *metodo Gasti* è in grado di classificare dieci diverse tipologie di impronte [Figura 1.5]:

- *adelta*, ulteriormente suddivisibili in quattro sotto tipologie
- *monodelta*, ulteriormente suddivisibili in tre sotto tipologie
- *bidelta*
- *composta*
- *tipologia "0"*, indicando un impronta imperfetta e/o mancante

Come è facile intuire, il metodo Gasti sfrutta la peculiarità delle impronte digitali chiamata delta, nome adottato per la forte somiglianza con l'omonima lettera dell'alfabeto greco Δ .

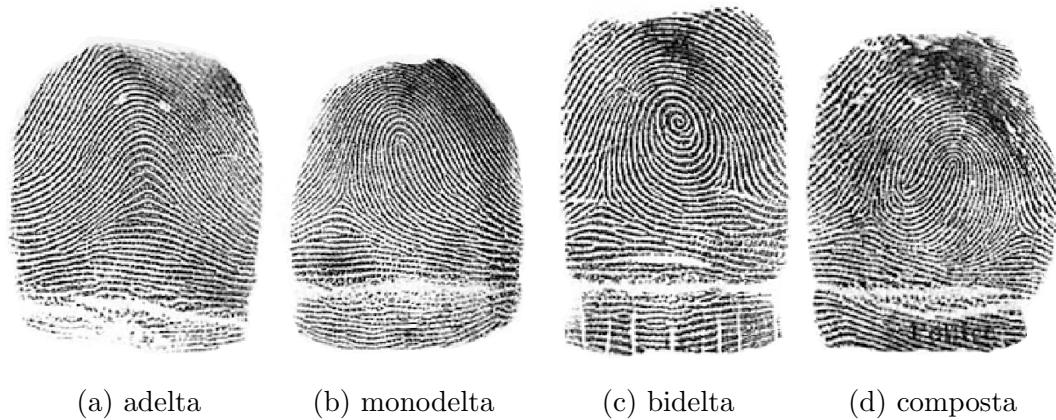


Figura 1.5: Tipologie di impronte del metodo Gasti

1.2.3 Minuzie

Nelle precedenti classificazioni sono state analizzate le impronte nella loro totalità, quindi andando a notare i pattern formati dalle *creste* su larga scala. Molto importante è comprendere come una *cresta* non sia mai totalmente priva di peculiarità, chiamate minuzie, le quali vanno a impreziosire l'unicità dell'impronta stessa. Negli anni è stata stilata una lista di sette minuzie le quali sono ritenute essere le più importanti ai fini dell'analisi forense [Figura 1.6]:

1. *Termination/Ending*: punto di terminazione di una cresta
2. *Bifurcation*: punto di divisione di una cresta in più creste
3. *Lake*: punto in cui una cresta biforca per riunirsi subito dopo
4. *Independent ridge*: cresta non connessa ad altre creste
5. *Point/Island*: cresta di dimensioni molto ridotte
6. *Spur*: cresta dalla quale si sviluppa una cresta *ending*
7. *Crossover*: cresta la quale funge da connessione tra due creste

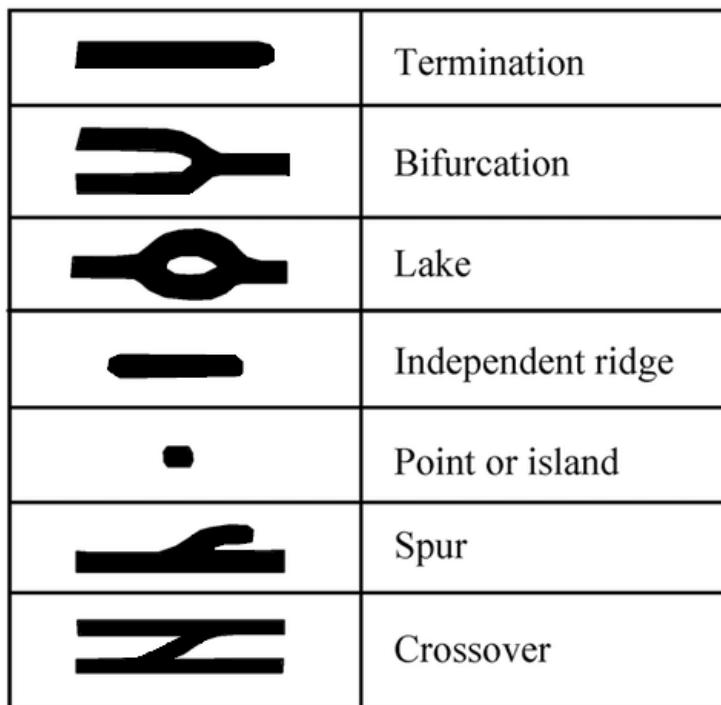


Figura 1.6: Lista di minuzie

1.3 Digitalizzazione delle impronte digitali

L'acquisizione delle immagini delle impronte digitali rappresenta la fase più critica in un sistema di autenticazione automatizzato. La qualità finale dell'immagine ottenuta ha un impatto significativo sulle prestazioni complessive del sistema, influenzandone direttamente l'efficacia e l'affidabilità.^[4]

Il metodo tradizionale per l'acquisizione delle impronte digitali prevede l'inchiostratura dei polpastrelli, seguita da un movimento di *rullatura* su carta per ottenere l'intero disegno dell'impronta. Un sistema più pratico, però, è rappresentato dai lettori ottici, anche se la qualità dell'immagine non è sempre ottimale a causa di vari fattori, come la formazione di condensa dovuta al contatto del dito con una superficie calda o la pressione esercitata.

Esistono diversi tipi di lettori di impronte digitali, che si possono suddividere in due categorie principali: *lettori a stato solido* e *lettori ottici*. In generale, la procedura di acquisizione consiste nel rotolare o appoggiare il dito su un'area di rilevamento che cattura la differenza tra le *crestae* e le *valli* dell'impronta. Tuttavia, il contatto del dito con una superficie solida può causare distorsioni e rumori nell'immagine acquisita.

Per ovviare a questi problemi, a partire dal 2010, sono stati sviluppati scanner di impronte digitali 3D senza contatto. Questi scanner adottano un approccio digitale, acquisendo informazioni tridimensionali dettagliate che permettono di riprodurre l'impronta con una risoluzione sufficientemente alta per registrare tutti i dettagli necessari, superando così le limitazioni del metodo tradizionale.

Per garantire la compatibilità tra diversi tipi di scanner, l'FBI ha stabilito delle specifiche nel 1999 per l'acquisizione delle immagini delle impronte digitali. Queste specifiche includono:

- *Risoluzione*: Lo standard è di 500 dpi (dots per inch), poiché tutti i sistemi di riconoscimento si basano sulle minuzie delle impronte. Per un dettaglio ultra-fine, la risoluzione standard sale a 1000 dpi.
- *Numero di pixel*: Determinato moltiplicando la risoluzione per l'area di rilevamento.
- *Quantizzazione dei livelli di grigio*: Indica il numero massimo di livelli di grigio nell'immagine di output ed è correlato al numero di bit usati per codificare l'intensità di ogni pixel. Ad esempio, con una risoluzione di 500 dpi e una codifica di 8 bit per pixel, si ottengono 256 livelli di grigio.

La digitalizzazione quindi porta inevitabilmente a dover trattare le impronte come *grafi*.

1.3.1 Grafi e Isomorfismi

Un *grafo* è una struttura matematica costituita da un insieme di nodi (o vertici) e un insieme di archi (o lati) che collegano coppie di nodi. Formalmente, un grafo G può essere definito come una coppia ordinata $G = \langle V, E \rangle$, dove:

- V è un insieme non vuoto i cui elementi sono chiamati vertici o nodi
- E è un insieme di coppie di elementi di V , chiamate archi o lati. Ogni arco $e \in E$ è rappresentato come una coppia $e = (u, v)$ dove $u, v \in V$.

Esistono diverse varianti di grafi, come i grafi diretti (dove gli archi hanno una direzione specifica) e i grafi non diretti (dove gli archi non hanno direzione). Altre caratteristiche possono includere pesi sugli archi, la presenza di loop (archi che connettono un vertice a se stesso) e la possibilità di grafi connessi o disconnessi.[2]

Per riuscire a contestualizzare i *grafi* nell'analisi di un'impronta digitale definiremo le *minuzie* come *nodi*, mentre definiremo le *creste* come *archi*.

Inoltre, i *grafi* da in considerazione saranno grafi non orientati, in quanto le *creste* non dispongono di un verso [Figura 1.7].

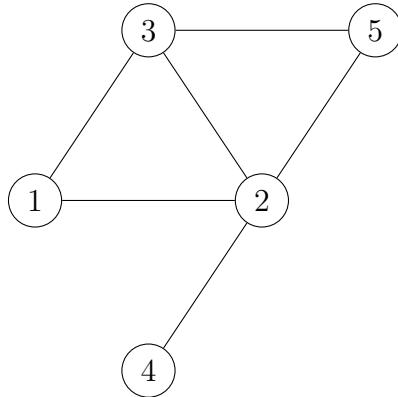


Figura 1.7: Esempio di grafo non orientato

Isomorfismo tra grafi

Una caratteristica interessante da cercare quando si parla di grafi in impronte digitali è l'*isomorfismo*.

Due grafi si dicono isomorfi se condividono la struttura degli archi seppur avendo i vertici in posizioni diverse.

Si parla quindi di isomorfismo se si riesce ad assegnare univocamente a ogni nodo del primo grafo un nodo del secondo preservandone gli archi. [Figura 1.8]

Formalmente:

Definizione 1.3.1 (Isomorfismo tra grafi).

Due grafi $G_1 = \langle V_1, E_1 \rangle$, $G_2 = \langle V_2, E_2 \rangle$ sono isomorfi se esiste una funzione biettiva $f : V(G_1) \longrightarrow V(G_2)$ tale che:

$$e_i = \{v_x, v_y\} \in E_1 \iff e_j = \{f(v_x), f(v_y)\} \in E_2$$

Si noti che la definizione 1.3.1 prevede che per ogni nodo del primo grafo, il relativo nodo nel secondo grafo ne preservi gli archi.

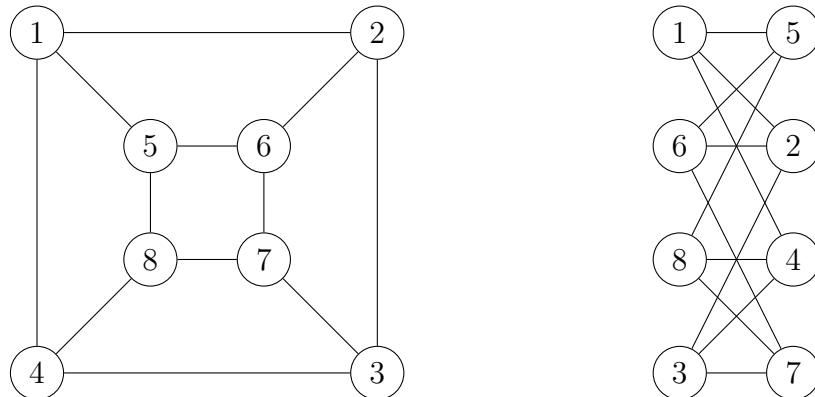


Figura 1.8: Esempio di isomorfismo tra grafi

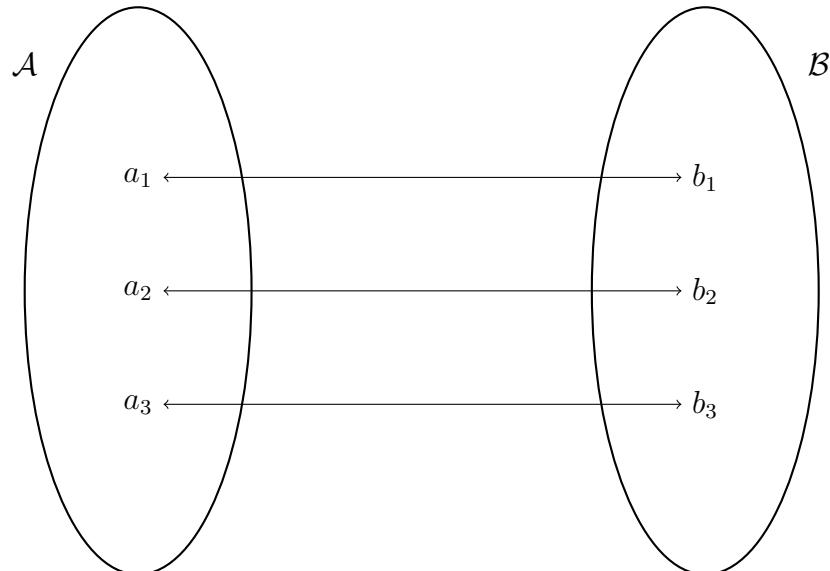


Figura 1.9: Esempio di Biezione

Sotto-isomorfismo tra grafi

Un'altra caratteristica dell'isomorfismo è il sotto-isomorfismo.

Quando si parla di sotto-isomorfismo, invece di considerare l'interezza di G_1 , si può considerare un sotto-grafo di G_1 , chiamato G'_1 e controllarne l'isomorfismo con G_2 [Figura 1.10]

Definizione 1.3.2 (Sotto-isomorfismo tra grafi).

Siano $G_1 = \langle V_1, E_1 \rangle$, $G_2 = \langle V_2, E_2 \rangle$ due grafi.

Il sotto-grafo $G'_1 = \langle V'_1 \subseteq V_1, E'_1 \subseteq E_1 \rangle$ è isomorfo a G_2 se $G'_1 \cong G_2$

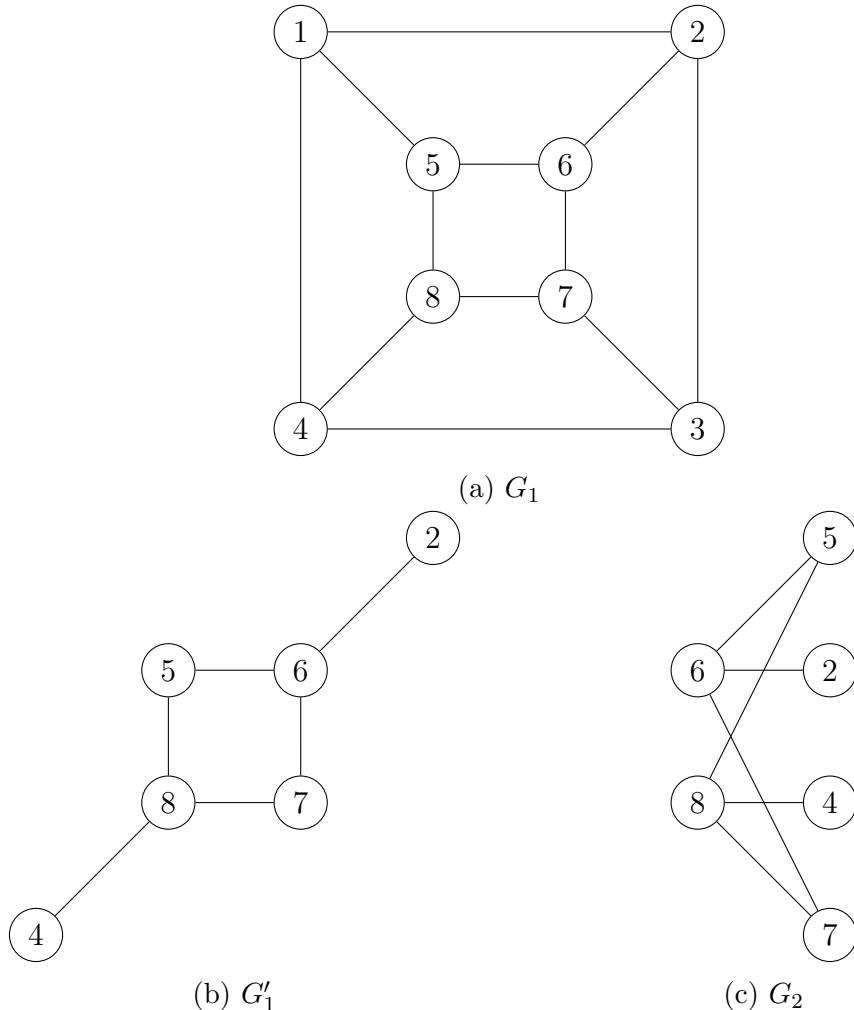


Figura 1.10: Esempio di sotto-isomorfismo tra grafi

1.4 Metodi di matching

Un'ampia gamma di algoritmi di matching per le impronte digitali è stata proposta nella letteratura. Tuttavia, la corrispondenza delle impronte digitali continua a rappresentare una sfida complessa, principalmente a causa delle difficoltà nel confrontare impronte digitali latenti, parziali e di bassa qualità. Gli approcci al matching delle impronte possono essere suddivisi in due principali categorie.

1. Match per minuzie
2. Match per features

Match per minuzie

Questo processo di matching consiste nel cercare un allineamento tra le minuzie memorizzate e quelle dell'impronta digitale di input, cercando di massimizzare il numero di abbinamenti tra le minuzie.

Data l'estrema improbabilità in cui un elevato numero di minuzie coincida casualmente tra due impronte digitali differenti, gli algoritmi basati su questa tecnica mirano a trovare il miglior allineamento possibile. Spesso, questi algoritmi considerano come corrispondenti coppie di minuzie anche se non coincidono perfettamente, ma sono abbastanza vicine da essere considerate un match.

Match per features

Quando un'impronta digitale è di bassa qualità, l'estrazione delle minuzie diventa spesso impraticabile. In queste condizioni, un'alternativa efficace consiste nell'estrarrre le caratteristiche dei pattern delle creste dell'impronta. Sebbene queste caratteristiche, come la forma delle creste, possano essere rilevate con maggiore affidabilità rispetto alle minuzie, la loro capacità di distinguere diverse impronte è generalmente inferiore.

1.5 AFIS

L'AFIS (Automated Fingerprint Identification System) è un sistema avanzato di hardware e software sviluppato per rispondere all'esigenza di ridurre i tempi tradizionalmente necessari per l'acquisizione e la catalogazione dei cartellini decadattilarli, nonché per rendere più rapide ed efficaci le ricerche di impronte digitali sconosciute all'interno di una banca dati informatizzata.

CAPITOLO 1. IMPRonte DIGITALI PER L'ANALISI FORENSE

Prima dell'introduzione dei computer e dell'AFIS, l'acquisizione e l'archiviazione delle impronte digitali erano processi manuali e laboriosi. Gli operatori di polizia utilizzavano inchiostro e cartellini di carta per rilevare le impronte digitali dei soggetti, registrando tutte e dieci le dita su cartellini decadattilari. Questi cartellini venivano poi catalogati manualmente e archiviati in grandi archivi fisici, dove erano suddivisi secondo vari criteri, come il tipo di disegno delle creste e forma. La ricerca di un'impronta specifica richiedeva un lungo lavoro di confronto visivo tra i cartellini fisici, una procedura lenta e suscettibile a errori.

Con l'introduzione dell'AFIS, questo processo è stato rivoluzionato. Le impronte digitali vengono ora acquisite in forma digitale e codificate attraverso un algoritmo avanzato, gestito dal sistema. Le ricerche possono essere eseguite su set completi di dieci impronte digitali, su frammenti di impronte o su impronte palmari, tutte rilevate dagli organi di polizia sulla scena del crimine. I terminali del sistema sono distribuiti presso le unità di polizia scientifica, come i RIS dei Carabinieri, e permettono la connessione via rete telematica alla Banca Dati del Casellario Centrale d'Identità del Servizio Polizia Scientifica. Questa banca dati centralizzata contiene informazioni biometriche utili per identificare i soggetti sia per finalità preventive che giudiziarie, rendendo il processo di identificazione più veloce, preciso e sicuro rispetto al passato.[5]

1.5.1 SourceAFIS

SourceAFIS è una libreria open-source progettata per l'elaborazione e il confronto di impronte digitali. Il suo scopo principale è identificare le persone confrontando impronte digitali catturate da diversi dispositivi biometrici. Originariamente sviluppata per il linguaggio di programmazione Java, la libreria è stata successivamente portata su altre piattaforme, come .NET.

Il funzionamento di SourceAFIS si basa su algoritmi di elaborazione delle immagini che estraggono i dettagli unici delle impronte digitali, chiamati minuzie. Queste minuzie includono caratteristiche come biforazioni, terminazioni e altre strutture presenti nelle creste delle impronte. Una volta estratte, le minuzie vengono confrontate con un set di impronte archiviate per determinare se c'è una corrispondenza.

La libreria è particolarmente utile in contesti in cui è necessario identificare o verificare l'identità di una persona basandosi su impronte digitali, come in applicazioni di sicurezza, controllo degli accessi o autenticazione personale. La natura open-source di SourceAFIS la rende accessibile a sviluppatori e ricercatori, che possono integrarla nelle loro applicazioni o modificarla per adattarla a esigenze specifiche.[6]

1.5.2 Funzionamento dell'algoritmo

L'algoritmo ideato da SourceAFIS può essere generalizzato e suddiviso in cinque parti:

1. Astrazione delle minuzie
2. Astrazione dei ridge
3. Creazione del template
4. Fase di match
5. Valutazione del match

Astrazione delle minuzie

Il primo passaggio nell'astrazione dei dati da dover confrontare, per fare ciò l'algoritmo divide l'astrazione delle minuzie in tre diverse fasi di preprocessing per poi inserire i dati in un template appositamente creato.[Figura 1.11]

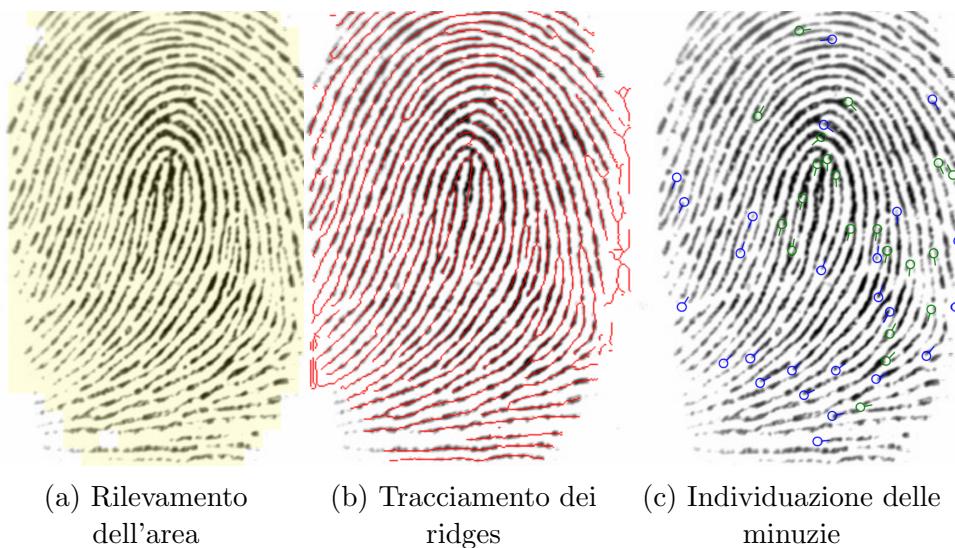


Figura 1.11: Estrazione delle minuzie di SourceAFIS

Astrazione degli archi

Dopo l'astrazione delle minuzie, c'è un ulteriore passo di astrazione, che produce gli archi. Un arco è una linea che collega due minuzie. L'arco ha una lunghezza e due angoli ereditati dalle sue minuzie. Gli angoli dell'arco sono espressi in modo relativo rispetto all'arco stesso. [Figura 1.12]

Queste tre proprietà dell'arco non cambiano quando l'arco viene spostato o ruotato, aspetto fondamentale per la correttezza del match.

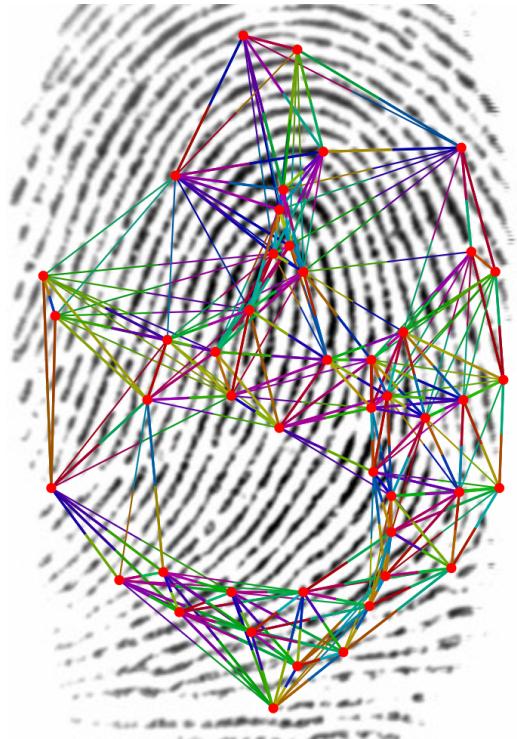


Figura 1.12: I colori sono determinati in base alla lunghezza e all'angolo degli archi. Archi simili hanno colori simili.

Fase di match

SourceAFIS cerca di trovare almeno un bordo condiviso dalle due impronte digitali che vengono confrontate. Questo viene fatto molto rapidamente utilizzando un algoritmo di nearest neighbor con prestazioni paragonabili a una tabella hash. Ciò ci fornirà la coppia radice, che è la coppia iniziale di minuzie corrispondenti, una per ciascuna impronta digitale.

A partire dalla coppia radice, SourceAFIS esplora i bordi verso l'esterno e costruisce un abbinamento composto da un certo numero di minuzie abbinate e bordi abbinati.



Figura 1.13: Il nodo minuzia è blu, L'albero di pairing è verde mentre il grafo che supporta i vertici è giallo.

Valutazione del match

SourceAFIS ora esamina attentamente l'abbinamento e decide se tale abbinamento rappresenta una corrispondenza o se è solo una coincidenza. Naturalmente, tutto potrebbe essere una coincidenza, ma la differenza tra una corrispondenza debole e una forte è che una corrispondenza forte è molto improbabile che sia una coincidenza.

È qui che SourceAFIS esegue la valutazione, l'ultima parte dell'algoritmo. L'idea di base è che ogni minuzia o arco abbinato è un evento che è improbabile che si verifichi casualmente. Più ci sono di questi eventi improbabili, meno è probabile che l'abbinamento sia solo una coincidenza. Quindi, l'algoritmo essenzialmente conta varie caratteristiche abbinate e le valuta anche in base a quanto strettamente corrispondono. La somma finale dei punteggi parziali viene adattata per allinearsi a una scala ragionevole e restituita dall'algoritmo. L'applicazione prende il punteggio e lo confronta con una soglia per decidere se si tratta di una corrispondenza o meno.

1.6 Ereditarietà delle impronte digitali

Una questione rilevante nel campo del riconoscimento delle impronte digitali riguarda l'ereditarietà delle loro caratteristiche. L'ereditarietà si riferisce alla trasmissione di determinati tratti dai genitori ai figli. È noto che caratteristiche come il colore dei capelli, della pelle e degli occhi siano ereditarie.

Negli studi sui dermatoglifi, la maggiore differenza genetica nelle impronte digitali si riscontra tra individui di etnie diverse. A ogni modo, anche persone non imparentate appartenenti alla stessa etnia mostrano pochissime somiglianze genetiche nelle impronte digitali. Tuttavia, esiste una certa somiglianza tra genitori e figli, poiché condividono metà del loro patrimonio genetico. Nei fratelli, la somiglianza è ancora maggiore, ma la somiglianza genetica massima si osserva nei gemelli monozigoti, i quali condividono lo stesso patrimonio genetico.

Nel 2019, *Anatomy* ha condotto uno studio in cui sono state prese in campione cinquanta famiglie per la quale sono state analizzate tutte le impronte effettuando dei test statistici.

I risultati di questo studio rafforzano l'argomento secondo cui le impronte digitali sono più determinate geneticamente che influenzate dall'ambiente, e che i modelli delle impronte vengono realmente trasmessi dai genitori ai figli. Tuttavia, il modo in cui vengono ereditati è piuttosto più complesso rispetto al semplice modello mendeliano o co-dominante.[1]

Capitolo 2

Dataset

In questo capitolo verrà trattato e approfondito il discorso sul dataset di impronte digitali utilizzato come argomento di questa tesi sotto tutti i suoi aspetti.

2.1 Creazione del dataset

Per garantire una partecipazione rappresentativa e attiva degli studenti, è stata progettata una strategia di coinvolgimento mirata. In primo luogo, è stata inviata un'email informativa a tutti gli studenti dell'Università, contenente dettagli chiari sull'opportunità di partecipare alla raccolta delle impronte digitali per la ricerca. L'email includeva un link diretto a un form online, accessibile tramite l'indirizzo email universitario di ciascun studente.

Il form online è stato progettato per essere semplice e intuitivo, permettendo agli studenti interessati di registrarsi rapidamente. Inoltre, il form consentiva di specificare il tipo di parente che avrebbero portato per partecipare alla raccolta delle impronte digitali presso la sede dei RIS di Parma. Questa personalizzazione ha facilitato la raccolta delle informazioni necessarie per organizzare correttamente le sessioni di raccolta.

Per raggiungere un pubblico più ampio, sono stati creati volantini promozionali con un codice QR che collegava direttamente al form online. Questo approccio ha ulteriormente semplificato il processo di registrazione, permettendo agli studenti di accedere rapidamente al form tramite i loro dispositivi mobili.

L'integrazione di questo sistema online ha reso più semplice la partecipazione e l'organizzazione dei dati, garantendo al contempo che le informazioni raccolte fossero complete e facilmente gestibili.

Una volta raccolti i dati degli studenti, sono state organizzate giornate specifiche in collaborazione con i RIS di Parma, durante le quali gli studenti e i

CAPITOLO 2. DATASET

loro familiari potevano recarsi alla sede per donare le proprie impronte digitali. Per la raccolta delle impronte è stato scelto il metodo della rullatura del dito su un cartellino con inchiostro, preferito per la sua capacità di catturare dettagli precisi e microscopici delle impronte digitali, come creste, solchi e delta. Questi dettagli, spesso fondamentali nell'analisi delle relazioni tra parenti stretti, potrebbero non essere altrettanto evidenti con altre tecniche, come le scansioni digitali.

In particolare, la rullatura con inchiostro è stata preferita per i seguenti vantaggi:

- *Dettagli microscopici*: Consente di catturare dettagli estremamente fini delle impronte digitali, che potrebbero non essere evidenti con altri metodi di acquisizione.
- *Conservazione dell'autenticità*: La rullatura con inchiostro offre un approccio più autentico e tradizionale, particolarmente rilevante in ambito forense.

2.2 Catalogazione delle impronte

Per garantire la riservatezza dei dati raccolti, pur consentendo una categorizzazione precisa delle impronte, è stato sviluppato un sistema di catalogazione avanzato e dettagliato.

Ogni gruppo familiare è stato identificato da un numero progressivo univoco, assegnato in sequenza a partire da "1". Inoltre, per preservare l'anonimato dell'individuo all'interno di ciascun gruppo familiare, sono state utilizzate sigle e codici che non rivelassero direttamente l'identità dei partecipanti.

- Gli studenti di riferimento sono stati contrassegnati dalla sigla "Ay" per i maschi e "Ax" per le femmine.
- Le sorelle sono state identificate con la lettera "E", mentre i fratelli con la lettera "D". In caso di più fratelli o sorelle, è stato aggiunto un numero progressivo.
- Nel caso speciale in cui i fratelli fossero gemelli, è stata utilizzata la lettera "G".
- Il padre è stato identificato con la lettera "B".
- La madre è stata identificata con la lettera "C".
- Il singolo dito è stato identificato da un numero progressivo partendo a contare dal pollice della mano destra al mignolo della sinistra.

- Nel caso fosse stata presa più volte l'impronta di uno stesso dito viene aggiunto un numero progressivo alla fine per indicare le diverse ripetizioni della stessa impronta.

Un esempio di etichetta attribuita all'impronta è: "01_G2_03_01", dove:

- "01" indica il primo gruppo familiare
- "G2" indica il gemello dello studente di riferimento
- "03" indica il dito medio della mano destra
- "01" indica la seconda acquisizione dell'impronta

2.3 Digitalizzazione delle impronte

Le impronte acquisite sono state digitalizzate e convertite in immagini *raster*, le quali sono poi state elaborate per estrarre le caratteristiche principali. Successivamente, queste informazioni sono state salvate in file di formato JSON, dove i dati strutturati sono stati ulteriormente processati. Da questi file sono stati estratti i grafici rappresentativi delle impronte, che verranno utilizzati come base per l'analisi e la ricerca.

2.4 Dimensione del dataset

Una volta acquisito e digitalizzato tutti i dati, il dataset comprende:

- 20 gruppi familiari
- 46 persone
- 460 dita
- 585 impronte

Questa mole di dati è abbastanza elevata per riuscire a condurre una studio di ricerca verosimile a una vera banca dati applicabile in un caso reale.

Capitolo 3

Algoritmi per la ricerca di sotto-isomorfismi

Una volta compreso a fondo l’idea dietro la struttura di un’impronta digitale, averne analizzato le sue caratteristiche e appreso i dati sui quali è possibile lavorare, può effettivamente iniziare lo studio di quello che sarà l’argomento cardine di questa tesi, ossia lo sviluppo di algoritmi volti alla ricerca di sotto-isomorfismi innovativi per riuscire ad attribuire un grado di somiglianza a due impronte senza effettivamente conoscerne la provenienza.

3.1 Preprocessing

Il primo passo da compiere per riuscire a effettuare delle ricerche attendibili su un’impronta è riuscire in qualche modo a ripulirla da tutti quei nodi creati dal rumore trovato in fase di acquisizione analogica dell’impronta. Con il termine *rumore* si intende l’insieme di nodi e archi rilevati in fase di parsing del dataset non provenienti realmente dall’impronta, ma rilevati dal software usato in fase di digitalizzazione il quale ha interpretato dall’immagine non completamente nitida delle creste fittizie non facenti parte dell’impronta originale.

Dopo un attento confronto tra un’impronta reale e i grafi ottenuti dal parsing del dataset è emerso come gran parte del rumore registrato si trovi in prossimità dei bordi dell’impronta.

Una volta appurato ciò, il primo passo è stato quello di pre-processare il grafo per eliminare tutti quei nodi presenti entro un certo offset dai nodi di bordo. Questa è l’unica soluzione adottabile senza conoscere a priori l’immagine reale dell’impronta in quanto, per motivi di privacy, è stato fornito soltanto il dataset senza poterne visionare l’originale. Tutte le implementazioni che verranno discusse nello sviluppo di questa tesi rimarranno succubi di

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

rumore il quale non si è riuscito a eliminare. L'implementazione è visionabile all'algoritmo 1.

Algoritmo 1 Preprocessing dei bordi

```
1: procedure REMOVEBORDERS(offset)
2:   for each outerNode in graph.node do
3:     range  $\leftarrow$  range(offset)
4:     border  $\leftarrow$  false
5:     for each innerNode in graph.node do
6:       if innerNode is inside(range) and innerNode is "border" then
7:         border  $\leftarrow$  true
8:         break
9:       if border is false then
10:        processedGraph.add(outerNode)
```

Successivamente, tramite una banale procedura, vengono ricollegati gli archi ai nuovi nodi eliminando gli archi divenuti obsoleti. Il risultato è il seguente:

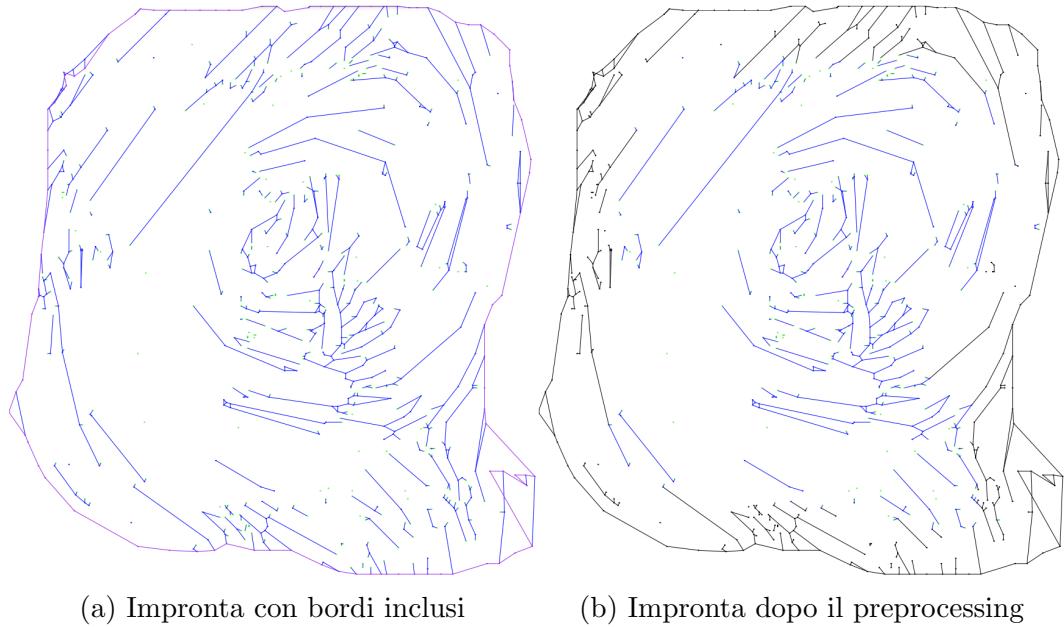


Figura 3.1: Impronta 01_Ay_01_00 prima e dopo il preprocessing, gli archi eliminati sono di colore nero

3.2 Ricerca tramite sotto-grafi

L'ideale sulla quale questa ricerca farà affidamento sarà diverso dalle metodologie di match le quali sono state approfondite nei capitoli precedenti. Gli algoritmi presentati non saranno basati sulla ricerca di un singolo grafo lungo l'intera impronta, ma saranno sviluppati invece per trovare sotto-grafi in zone precise dell'impronta, in quanto la supposizione fatta è quella di voler trovare molti pattern simili nelle stesse zone di due impronte diverse per poter poi deciderne successivamente la similarità.

Un problema immediatamente sorto riguarda la complessità del problema nella sua generalità, poiché la ricerca di sotto-isomorfismi è nota per aver una complessità computazionale NP-Completa. Questo significa che non è possibile risolvere il problema in un tempo polinomiale.

Tuttavia, l'obiettivo finale è quello di trovare similarità in una zona precisa dell'impronta stessa, perciò seguendo la supposizione posta come preambolo al problema l'idea è stata quella di sfruttare una *box* di dimensione modulabile per delimitare l'area di ricerca.

Utilizzando questo metodo, i sotto-grafi che verranno trovati avranno una cardinalità contenuta rispetto alla totalità dei nodi e archi presenti nell'intera impronta. Questo porta oltre a una semplificazione computazionale anche il vantaggio di riuscire a trovare un sotto-grafo in una zona effettivamente ristretta rispetto all'impronta stessa, esattamente come la supposizione sulla quale viene fatto affidamento si basa. Di seguito l'algoritmo 2 mostra come viene per l'appunto implementata la *box* prima descritta per la creazione del sotto-grafo.

Algoritmo 2 Esempio di implementazione della box tramite il suo range

```
1: procedure SUBGRAPH(graph,range)
2:   for each node in graph.node do
3:     if node is inside(range) then
4:       subgraph.addEdge(from, to)
5:       subgraph.addNode(node)
```

Una volta sviluppata la *box* non ci rimane che trovare una strategia efficace per la creazione di sotto-grafi isomorfi tra loro. Per fare ciò l'idea è stata quella di provare con due diversi approcci i quali potrebbero fare al caso nostro: DFS e BFS.

CAPITOLO 3. ALGORITMI PER LA RICERCA DI
SOTTO-ISOMORFISMI

3.2.1 DFS

La prima strategia a esser illustrata è la ricerca di sotto-grafi tramite l'algoritmo DFS (Depth First Search).

L'algoritmo DFS è una tecnica utilizzata per esplorare o cercare in modo sistematico tutti i vertici e gli spigoli di un grafo, sia esso orientato o non orientato. DFS procede esplorando in profondità ciascun ramo del grafo prima di retrocedere, tornando indietro solo quando raggiunge un vicolo cieco, cioè un nodo senza ulteriori nodi adiacenti non visitati. Questa strategia consente di attraversare l'intero grafo partendo da un nodo iniziale, spostandosi lungo i percorsi disponibili fino a che tutti i nodi connessi siano stati visitati. Utilizza quindi un approccio LIFO (*Last In First Out*).

Nel pratico questo approccio consente di trovare quindi nella maggior parte dei casi un sotto-grafo di forma *lineare*, ovvero un sotto-grafo il quale segue a tutti gli effetti la forma della cresta. L'algoritmo 3 mostra l'implementazione in pseudo codice mentre la Figura 3.2 mostra un esempio di albero DFS.

Algoritmo 3 Depth-First Search (DFS)

```

1: procedure DFS( $G, v$ ) ▷  $G$  è il grafo,  $v$  è il nodo iniziale
2:   mark  $v$  as visited
3:   for each  $w$  adjacent to  $v$  do
4:     if  $w$  is not visited then
5:       DFS( $G, w$ )

```

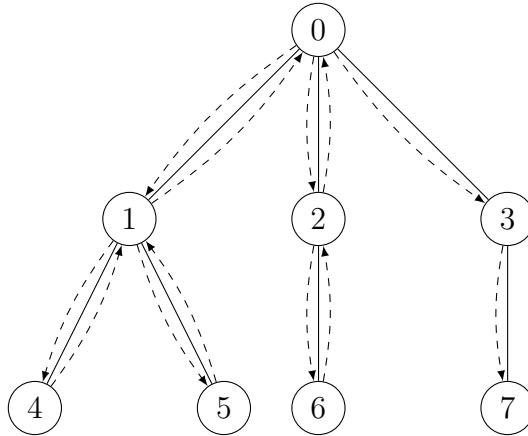


Figura 3.2: Esempio di albero DFS.
L'output è il seguente : 0, 1, 4, 5, 2, 6, 3, 7

Implementazione DFS

Una volta introdotto l'algoritmo DFS nella sua generalità, è possibile implementarlo per la creazione del sotto-grafo. Per fare ciò viene analizzato ogni nodo presente nell'impronta e posizionata la box con centro sul nodo in questione. Partendo dal nodo in analisi viene sviluppato il grafo seguendo la logica DFS fino a trovare un sotto-grafo di cardinalità fissata scelta in fase di test. Una volta aggiunti tutti i nodi necessari alla struttura del sotto-grafo, tramite una semplice procedura vengono aggiunti tutti gli archi correlati ai nodi presenti per mantenere una solida struttura.

In figura 3.3 viene mostrato l'insieme di sotto-grafi riscontrati nell'impronta 01_Ay_01_00 con cardinalità pari a cinque, mentre in figura 3.4 viene mostrato un ingrandimento di un sotto-grafo all'interno della propria box.



Figura 3.3: Impronta 01_Ay_01_00 dopo aver eseguito l'algoritmo DFS

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

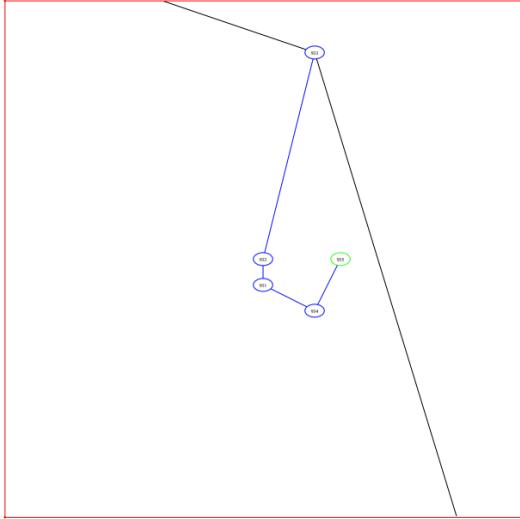


Figura 3.4: Box contenente un sotto-grafo trovato tramite l'algoritmo DFS

3.2.2 BFS

La seconda strategia che verrà illustrata riguarda la ricerca di sotto-grafi mediante l'algoritmo BFS (Breadth-First Search). Questo algoritmo rappresenta una tecnica fondamentale per l'esplorazione e la ricerca all'interno di strutture complesse come grafi e alberi.

BFS si basa sull'utilizzo di una coda (*queue*) come struttura dati principale per gestire l'ordine di visita dei nodi. A differenza di altre tecniche di ricerca, BFS esplora i nodi di un grafo in modo sistematico, procedendo per livelli. In altre parole, l'algoritmo parte dal nodo iniziale e, prima di esplorare nodi più distanti, visita tutti i nodi adiacenti al nodo di partenza. Questo processo di esplorazione garantisce che ogni nodo venga visitato nella sua distanza minima dal nodo iniziale, evitando così di tralasciare nodi potenzialmente rilevanti in una prima fase di esplorazione.

L'algoritmo BFS adotta un approccio FIFO (*First In First Out*), che consente di esplorare un livello alla volta prima di passare al successivo, espandendo l'area di ricerca in ampiezza piuttosto che in profondità. Grazie a questa caratteristica, BFS è particolarmente efficace nel trovare sotto-grafi tendenti ad avere una forma più articolata e complessa. In effetti, proprio per la sua capacità di espandersi orizzontalmente, BFS può rivelare strutture nascoste all'interno del grafo le quali potrebbero non essere evidenti con altre tecniche di esplorazione.

L'algoritmo 4, riportato di seguito, presenta l'implementazione in pseudo codice dell'algoritmo BFS, mentre la Figura 3.5 fornisce un esempio visivo di un albero esplorato mediante questa tecnica.

Algoritmo 4 Breadth-First Search (BFS)

```

1: procedure BFS( $G, s$ )            $\triangleright G$  è il grafo,  $s$  è il nodo sorgente
2:   initialize all nodes  $u$  with  $\text{color}[u] \leftarrow \text{white}$ ,  $\mathbf{d}[u] \leftarrow \infty$ ,  $\mathbf{p}[u] \leftarrow \text{NIL}$ 
3:    $\text{color}[s] \leftarrow \text{gray}$ 
4:    $\mathbf{d}[s] \leftarrow 0$ 
5:    $\mathbf{p}[s] \leftarrow \text{NIL}$ 
6:   initialize the queue  $Q$  empty
7:   enqueue  $s$  in  $Q$ 
8:   while  $Q$  is not empty do           $\triangleright Q$  non è vuota
9:     dequeue node  $u$  from the queue  $Q$   $\triangleright$  estraie il nodo  $u$  dalla coda  $Q$ 
10:    for each node  $v$  adjacent to  $u$  do     $\triangleright$  ogni nodo  $v$  adiacente a  $u$ 
11:      if  $\text{color}[v]$  is white then            $\triangleright$  colore di  $v$  è bianco
12:         $\text{color}[v] \leftarrow \text{gray}$ 
13:         $\mathbf{d}[v] \leftarrow \mathbf{d}[u] + 1$ 
14:         $\mathbf{p}[v] \leftarrow u$ 
15:        enqueue  $v$  in the queue  $Q$         $\triangleright$  inserisci  $v$  nella coda  $Q$ 
16:    $\text{color}[u] \leftarrow \text{black}$ 

```

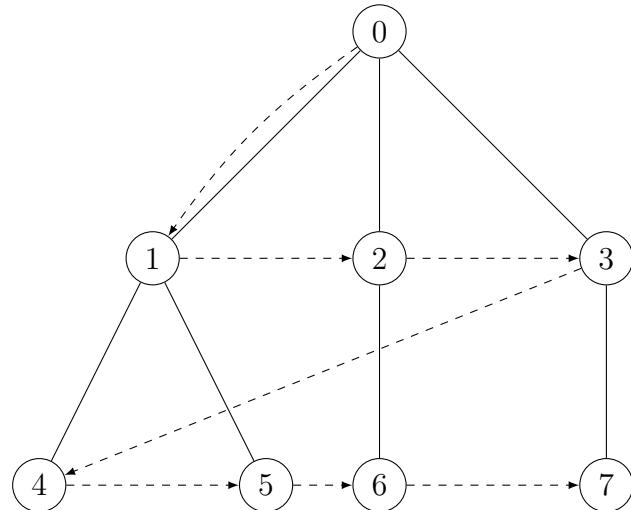


Figura 3.5: Esempio di albero BFS.
L'output è il seguente : 0, 1, 2, 3, 4, 5, 6, 7

Implementazione BFS

Una volta introdotto l'algoritmo BFS nella sua generalità, come è stato per DFS, è possibile implementarlo per la creazione del sotto-grafo. Il processo di creazione è il medesimo utilizzato in precedenza, con unica differenza il metodo di esplorazione, il quale non sarà più per profondità ma per l'appunto in ampiezza.

In figura 3.6 viene mostrato l'insieme di sotto-grafi riscontrati nell'impronta 01_Ay_01_00 con cardinalità pari a cinque, mentre in figura 3.7 viene mostrato un ingrandimento di un sotto-grafo all'interno della propria box.

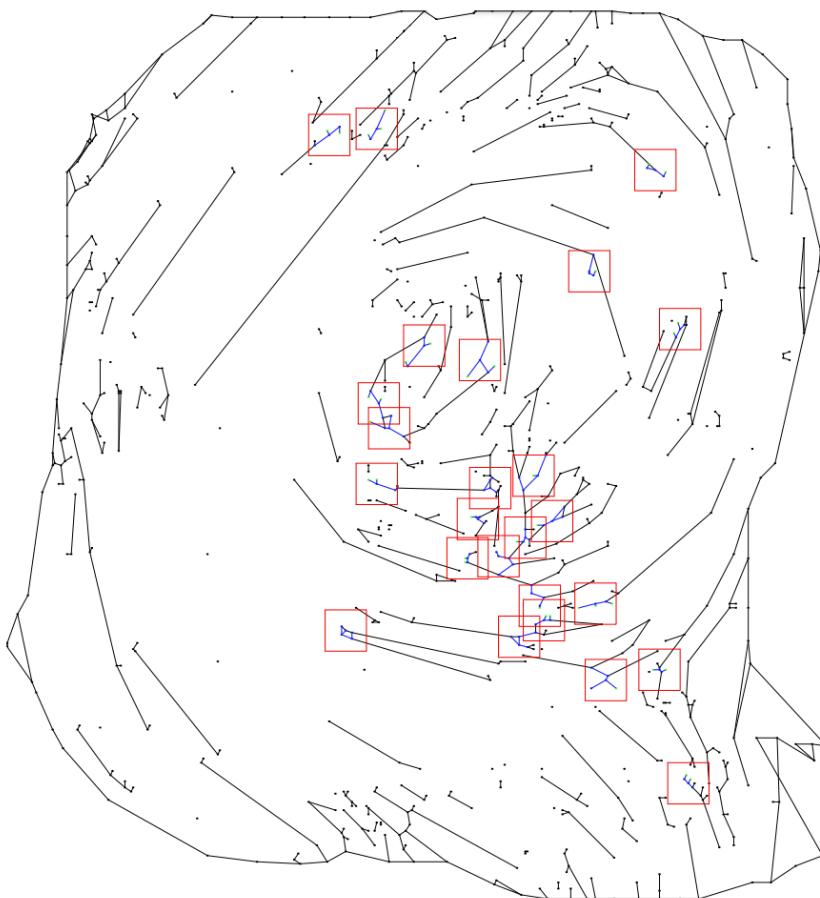


Figura 3.6: Impronta 01_Ay_01_00 dopo aver eseguito l'algoritmo BFS

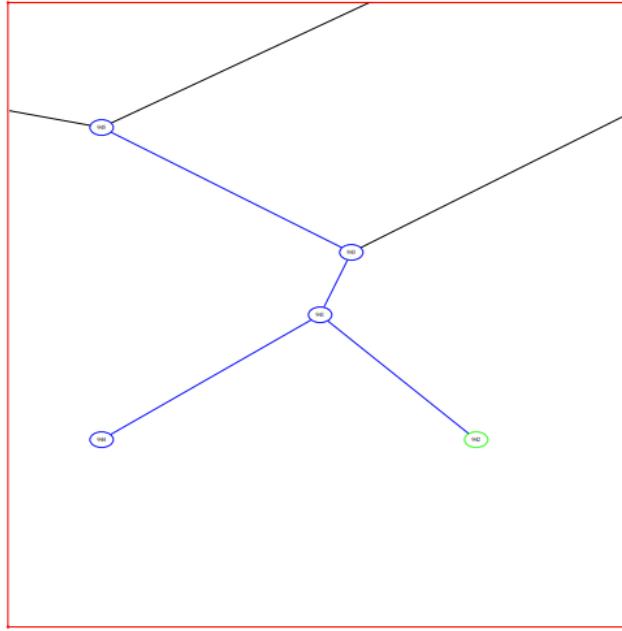


Figura 3.7: Box contenente un sotto-grafo trovato tramite l'algoritmo BFS

3.2.3 Controllo dell'isomorfismo

Una volta ottenuti i sotto-grafi, la parte cruciale del processo consiste nel verificare se le varie coppie identificate siano effettivamente simili tra loro. A tal fine, è stato necessario sviluppare un metodo efficace per accoppiare i diversi sotto-grafi. La tecnica utilizzata si basa sul principio con cui sono stati creati gli stessi sotto-grafi: l'idea è di imporre una box di dimensioni generalmente maggiori, solitamente doppie rispetto a quelle utilizzate per la creazione dei sotto-grafi, al fine di aggirare eventuali problematiche legate alla rotazione dell'immagine, che potrebbero non essere risolvibili altrimenti. Questa box viene centrata sul nodo centrale del sotto-grafo considerato. Per individuare il nodo centrale del sotto-grafo, è stato adottato un criterio basato sul calcolo del valore minimo della somma delle distanze euclidee tra ciascun nodo del sotto-grafo e tutti gli altri nodi, utilizzando le formule 3.1 e 3.2:

$$S(\mathbf{n}_i) = \sum_{\mathbf{n}_j \in \text{nodi}} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3.1)$$

$$\mathbf{n}_{\min} = \arg \min_{\mathbf{n}_i \in \text{nodi}} S(\mathbf{n}_i) \quad (3.2)$$

Dopo aver definito la regione di accettazione dei sotto-grafi, il passo successivo consiste nel verificare concretamente se la coppia di grafi sia effettivamente isomorfa. Per affrontare questo problema, si è scelto di utilizzare gli alberi.

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

Questa scelta è motivata dal fatto che, attraverso l'uso degli alberi, risulta più semplice e diretto verificarne l'isomorfismo, poiché i cicli che potrebbero essersi formati durante la costruzione del sotto-grafo vengono eliminati, semplificando così la struttura e facilitando il confronto tra le coppie di grafi.

L'implementazione degli alberi è stata realizzata utilizzando l'algoritmo di Depth-First Search (DFS). Questa scelta si basa sulla particolare conformazione dei grafi comunemente riscontrati in fase di sviluppo. È fondamentale sotto-lineare che, indipendentemente dal metodo di costruzione dell'albero adottato, i risultati finali restano invariati. Questo perché, per garantire coerenza nella struttura degli alberi, le coppie di nodi vengono organizzate utilizzando come radice il nodo centrale identificato tramite le formule 3.1 e 3.2.

Per determinare l'isomorfismo tra alberi, è essenziale introdurre un metodo per ordinare tali strutture. A tal fine, è stata sviluppata una tecnica che si basa sull'ordinamento delle stringhe generate tramite una specifica funzione di hash. Questa funzione di hash, che verrà utilizzata per rappresentare i nodi dell'albero come stringhe, è definita nel seguente modo:

$$h(n) \begin{cases} "(e)" & \text{se } n \text{ è un nodo foglia di tipo ending} \\ "(b)" & \text{se } n \text{ è un nodo foglia di tipo bifurcation} \\ "(e" + h(n_1) + \dots + h(n_n) + ")" & \text{se } n \text{ è un nodo padre di tipo ending} \\ "(b" + h(n_1) + \dots + h(n_n) + ")" & \text{se } n \text{ è un nodo padre di tipo bifurcation} \end{cases}$$

In altre parole, la funzione $h(n)$ genera una stringa univoca per ciascun nodo, che tiene conto del tipo del nodo stesso (ovvero se è una foglia o un nodo padre e se è di tipo *ending* o *bifurcation*) e delle stringhe associate ai suoi figli, se presenti. Questa rappresentazione consente di confrontare e ordinare i nodi dell'albero in modo univoco.

Una volta definita la funzione di hash $h(n)$, si procede ordinando lessicograficamente le stringhe ottenute per i vari nodi. Questo permette di ottenere una rappresentazione ordinata dei figli di ogni nodo all'interno dell'albero. Data questa rappresentazione, si possono confrontare due alberi t_1 e t_2 in modo diretto:

$$t_1 \prec t_2 \iff h(t_1) \prec h(t_2)$$

Questa tecnica permette di sfruttare l'ordinamento delle stringhe per ordinare i figli di ciascun nodo in modo tale che, se due alberi siano isomorfi, le loro rispettive stringhe risultino essere identiche. In tal modo, la verifica dell'isomorfismo tra alberi viene ridotta al confronto delle stringhe ottenute, rendendo il processo più efficiente e sistematico.

A seguire, viene presentato lo pseudo codice dell'algoritmo sviluppato per la creazione degli alberi DFS e l'algoritmo dedicato al controllo dell'isomorfismo degli alberi, rispettivamente Algoritmo 5 e Algoritmo 6.

Algoritmo 5 Creazione albero DFS

```

1: function DFS
2:   map  $\leftarrow$  new HashMap()
3:   for each node n in nodes do
4:     n.color  $\leftarrow$  white                                 $\triangleright$  Initialize DFS colors
5:     root  $\leftarrow$  new TreeNode(center.coordinates, center.type, center.index)
6:     DFSVISIT(root, center, color)
7:   return new Tree(root)
8: function DFSVISIT(treeNode, graphNode, colorMap)
9:   graphNode.color  $\leftarrow$  grey
10:  for each edge e in edges do
11:    if (e.from = graphNode and e.to.color = white)
12:    or (e.to = graphNode and e.from.color = white) then
13:      if e.from = graphNode then
14:        n  $\leftarrow$  e.to
15:      else
16:        n  $\leftarrow$  e.from
17:      tmp  $\leftarrow$  new TreeNode(n.coordinates, n.type, n.index)
18:      treeNode.addChild(tmp)
19:      DFSVISIT(tmp, n, color)
20:   graphNode.color  $\leftarrow$  black

```

Algoritmo 6 Isomorfismo tra alberi tramite ordinamento di stringHash

```

1: function ISOMORPH(Tree other)
2:   return root.stringHash() = other.root.stringHash()
3: function STRINGHASH
4:   string  $\leftarrow$  "(" + CHARAT(type, 0)
5:   childrenStringHash  $\leftarrow$  {}                                 $\triangleright$  Lista di stringhe
6:   for each nodo n in children do
7:     hash  $\leftarrow$  n.STRINGHASH
8:     APPEND(childrenStringHash, hash)
9:   SORT(childrenStringHash)
10:  for each string hash in childrenStringHash do
11:    string  $\leftarrow$  string + hash
12:  string  $\leftarrow$  string + ")"
13:  return string

```

CAPITOLO 3. ALGORITMI PER LA RICERCA DI
SOTTO-ISOMORFISMI

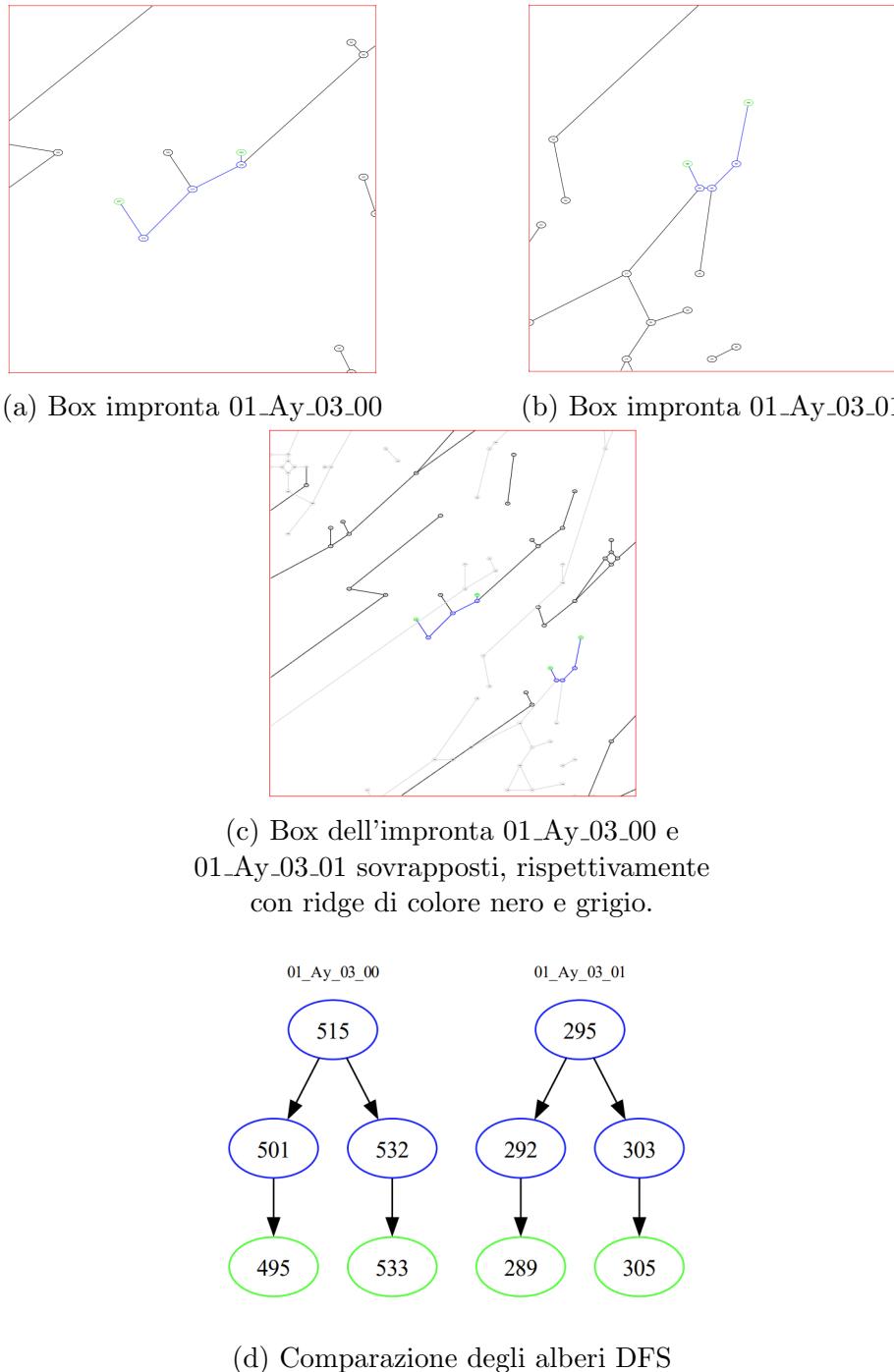
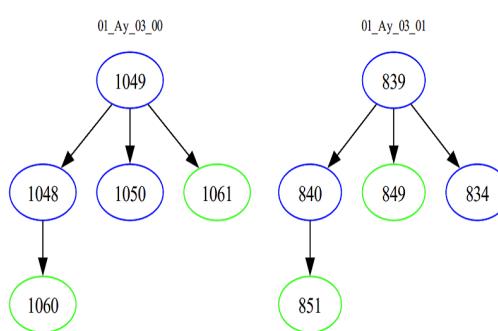
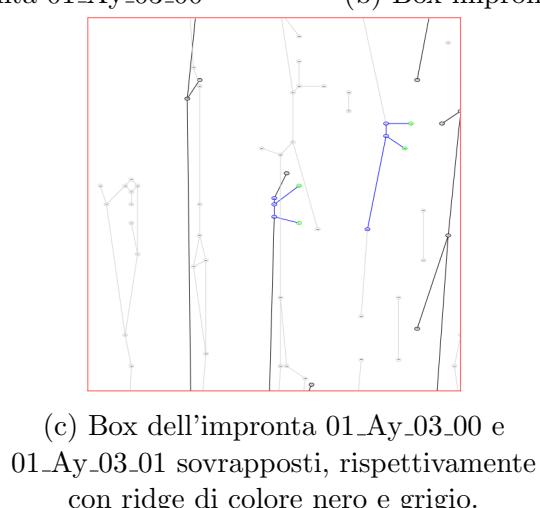
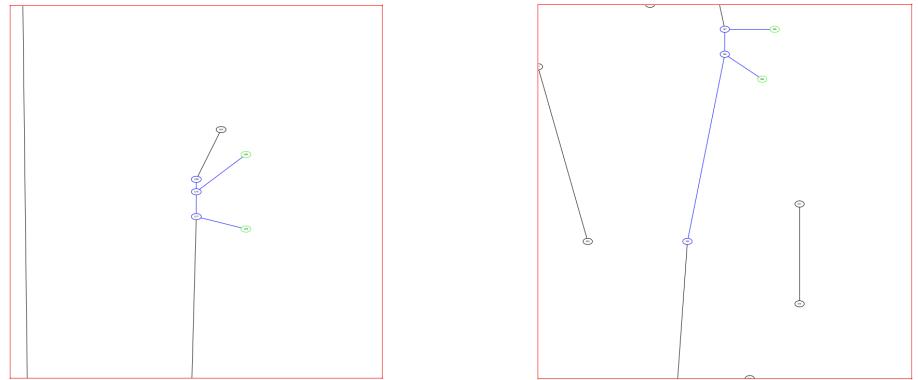


Figura 3.8: Esempio di isomorfismo trovato tramite DFS

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI



(d) Comparazione degli alberi DFS

Figura 3.9: Esempio di isomorfismo trovato tramite BFS

3.3 Ricerca tramite nodi

In seguito allo sviluppo di algoritmi avanzati per la comparazione di strutture di modeste dimensioni, costituite da nodi e archi, è emerso un problema significativo che si è rivelato difficilmente risolvibile. Questo problema riguarda l’incapacità di controllare in modo arbitrario l’orientamento di questi sotto-grafi all’interno delle strutture esaminate. Nonostante gli algoritmi fossero in grado di preservare le proprietà fondamentali dell’isomorfismo tra le strutture, si è constatato che per ottenere una corrispondenza perfetta, sarebbe stato necessario che le due strutture isomorfe condividessero non solo le stesse proprietà, ma anche la medesima configurazione e orientamento all’interno del grafo. Di conseguenza, per avvicinarsi a questo obiettivo, è stata adottata una strategia alternativa che prevede l’analisi esclusiva dei nodi del grafo, escludendo deliberatamente la considerazione degli archi.

3.3.1 Analisi del singolo nodo

La prima strategia adottata è stata quella di analizzare un singolo nodo del grafo per volta. In particolare, l’idea è stata quella di restringere lo spazio di analisi, riducendo la dimensione della box di creazione a un limite minimo, tale da contenere unicamente un singolo nodo. Questa riduzione impedisce lo sviluppo di un sotto-grafo completo, ma permette di ottenere una serie di sotto-grafi elementari, ciascuno dei quali contenente un solo nodo. Così facendo, si genera un numero di sotto-grafi pari alla cardinalità del grafo originale.

Una volta ottenuti questi sotto-grafi elementari, la strategia consiste nel confrontarli con i sotto-grafi corrispondenti estratti dall’impronta da analizzare. Questo confronto è mirato a identificare un match basato esclusivamente sulla posizione esatta dei nodi, tenendo conto della loro tipologia specifica. In altre parole, l’obiettivo è verificare se ciascun nodo, con le sue caratteristiche uniche, occupi la stessa posizione relativa in entrambi i grafi. Questa tecnica permette di ottenere un’analisi molto precisa, concentrata sull’allineamento esatto dei nodi, fornendo una base solida per valutare la somiglianza tra i due grafi considerati.

Per ovviare a eventuali errori dovuti alla leggera rotazione dell’immagine in fase di digitalizzazione, la soluzione è stata ingrandire di qualche pixel la box di comparazione, mantenendo il confronto dei singoli nodi andando a mantenere la biunivocità del match tramite la seguente implementazione visionabile all’algoritmo 7, inoltre viene fornita una dimostrazione visiva sui due passaggi effettuati per trovare i nodi di match, rispettivamente in figura 3.10 e in figura 3.11.

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

Algoritmo 7 Match dei singoli nodi

```
1: function NODESCOMPARE(Graph other)
2:   if node1 is inside(range) & node2 is inside(range) then
3:     if node1.type = node2.type then
4:       result.add(node1)
5:       result.add(node2)
6:     other.remove(node2)
```



Figura 3.10: Mappatura di ogni nodo dopo la fase di preprocessor

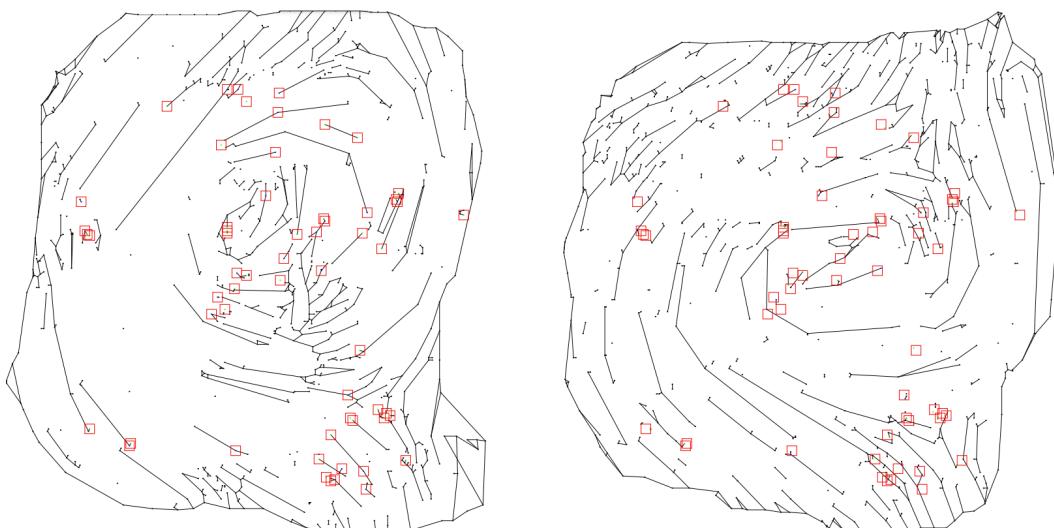


Figura 3.11: Riscontro dei nodi di match

3.3.2 Analisi di nodi in una matrice

Questa seconda strategia, basata anch'essa sul confronto tra nodi, si fonda sull'idea di utilizzare una matrice a dimensione modulabile per analizzare e confrontare le impronte digitali. L'approccio non si limita a considerare le zone ricche di nodi, ma attribuisce una rilevanza anche alle aree prive di minuzie, offrendo così un'analisi più completa e dettagliata.

L'algoritmo prevede di suddividere l'impronta in una griglia, dove ogni cella della matrice rappresenta una porzione dell'area complessiva. In ciascuna di queste celle verrà eseguito un controllo per mappare il numero e il tipo di nodi presenti, fornendo così un quadro dettagliato della distribuzione delle caratteristiche dell'impronta stessa. Il processo viene poi ripetuto per entrambe le impronte che devono essere confrontate, e le celle corrispondenti, ovvero quelle che occupano la stessa posizione in entrambe le matrici, vengono messe a confronto.

Se una cella risulta vuota in entrambe le matrici, cioè non contiene nodi, verrà etichettata come tale e di conseguenza le verrà attribuito un peso diverso in fase di calcolo del punteggio finale, questo per dare per l'appunto una valore anche a zone prive di nodi.

L'algoritmo

L'algoritmo subisce una trasformazione sostanziale rispetto alle versioni presentate nei capitoli precedenti. In particolare, l'idea delle box viene abbandonata per fare spazio a un nuovo approccio basato su aree di ricerca che sono state preventivamente mappate e rese statiche nell'analisi. Questo rappresenta una differenza significativa rispetto alle box, che venivano generate in modo dinamico attorno a ciascun nodo. Il nuovo approccio permette una maggiore stabilità e coerenza nelle fasi di analisi. L'algoritmo, dunque, viene riorganizzato e suddiviso nei seguenti passaggi:

1. Definizione della dimensione e creazione della matrice
2. Mappatura di ogni nodo nella cella corrispondente

Creazione della matrice

Poiché la matrice è un elemento statico e viene creata prima di effettuare il confronto, è fondamentale garantire che le sue dimensioni siano sufficientemente ampie per contenere ogni nodo di entrambe le impronte. Una matrice sottodimensionata non riuscirebbe a raccogliere correttamente tutti i nodi, compromettendo l'accuratezza del confronto tra le impronte. Lo pseudo codice 8 mostra come avviene la corretta definizione dei valori esterni della matrice,

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

mentre lo pseudo codice 9 descrive la creazione di ogni cella della matrice, partendo dalla cella in basso a sinistra per arrivare alla cella in alto a destra.

Algoritmo 8 Definizione della dimensione massima della matrice

```
1: Inizializza: maxX, minX, maxY, minY
2: for each node  $n$  in  $g1.nodes$  & node  $n$  in  $g2.nodes$  do
3:   if  $n.getX() > maxX$  then
4:      $maxX \leftarrow n.getX()$ 
5:   if  $n.getX() < minX$  then
6:      $minX \leftarrow n.getX()$ 
7:   if  $n.getY() > maxY$  then
8:      $maxY \leftarrow n.getY()$ 
9:   if  $n.getY() < minY$  then
10:     $minY \leftarrow n.getY()$ 
```

Algoritmo 9 Creazione delle celle della matrice

```
1: Input:  $maxX, minX, maxY, minY, colonne, righe$ 
2: Output:  $boxArray$ 
3:  $segmentX \leftarrow \frac{(maxX - minX)}{colonne}$ 
4:  $segmentY \leftarrow \frac{(maxY - minY)}{righe}$ 
5:  $nuovoMinY \leftarrow minY$ 
6:  $nuovoMaxY \leftarrow minY + segmentY$ 
7: for  $i \leftarrow 0$  to  $righe$  do
8:    $nuovoMinX \leftarrow minX$ 
9:    $nuovoMaxX \leftarrow minX + segmentX$ 
10:  for  $j \leftarrow 0$  to  $colonne$  do
11:     $index \leftarrow i \times colonne + j$ 
12:     $boxArray.add(Box(index, nuovoMinX, nuovoMaxX, nuovoMinY,$ 
13:       $nuovoMaxY))$ 
14:     $nuovoMinX \leftarrow nuovoMaxX$ 
15:     $nuovoMaxX \leftarrow nuovoMaxX + segmentX$ 
16:     $nuovoMinY \leftarrow nuovoMaxY$ 
17:     $nuovoMaxY \leftarrow nuovoMaxY + segmentY$ 
```

6	7	8
3	4	5
0	1	2

Figura 3.12: Mappatura degli indici della matrice

Mappatura dei nodi

Una volta definita la matrice, il prossimo passo è quello di assegnare ogni nodo dell’impronta a una precisa cella della matrice. Per fare ciò è possibile servirci della seguente formula per calcolare l’indice della cella:

$$\lfloor (y_{nodo} - \min Y) \times \frac{\text{righe}}{\max Y - \min Y} \rfloor \times \text{colonne} + \lfloor (x_{nodo} - \min X) \times \frac{\text{colonne}}{\max X - \min X} \rfloor$$

Una volta trovato l’indice corretto, basterà analizzare il tipo di nodo in questione e salvare il risultato nell’area di memoria appositamente allocata per gestire le occorrenze, come segue nello pseudo codice 10:

Algoritmo 10 Ricerca indice e inserimento

```

1: for each node n in graph.nodes do
2:   colonna  $\leftarrow \left\lfloor \frac{n.getX() - \min X}{\text{segmentX}} \right\rfloor$ 
3:   riga  $\leftarrow \left\lfloor \frac{n.getY() - \min Y}{\text{segmentY}} \right\rfloor$ 
4:   index  $\leftarrow \text{riga} \times \text{colonne} + \text{colonna}$ 
5:   if n.type = bifurcation then
6:     boxArray.get(index).increaseBifurcationCounterG1()
7:   else if n.type = ending then
8:     boxArray.get(index).increaseEndingCounterG1()

```

CAPITOLO 3. ALGORITMI PER LA RICERCA DI SOTTO-ISOMORFISMI

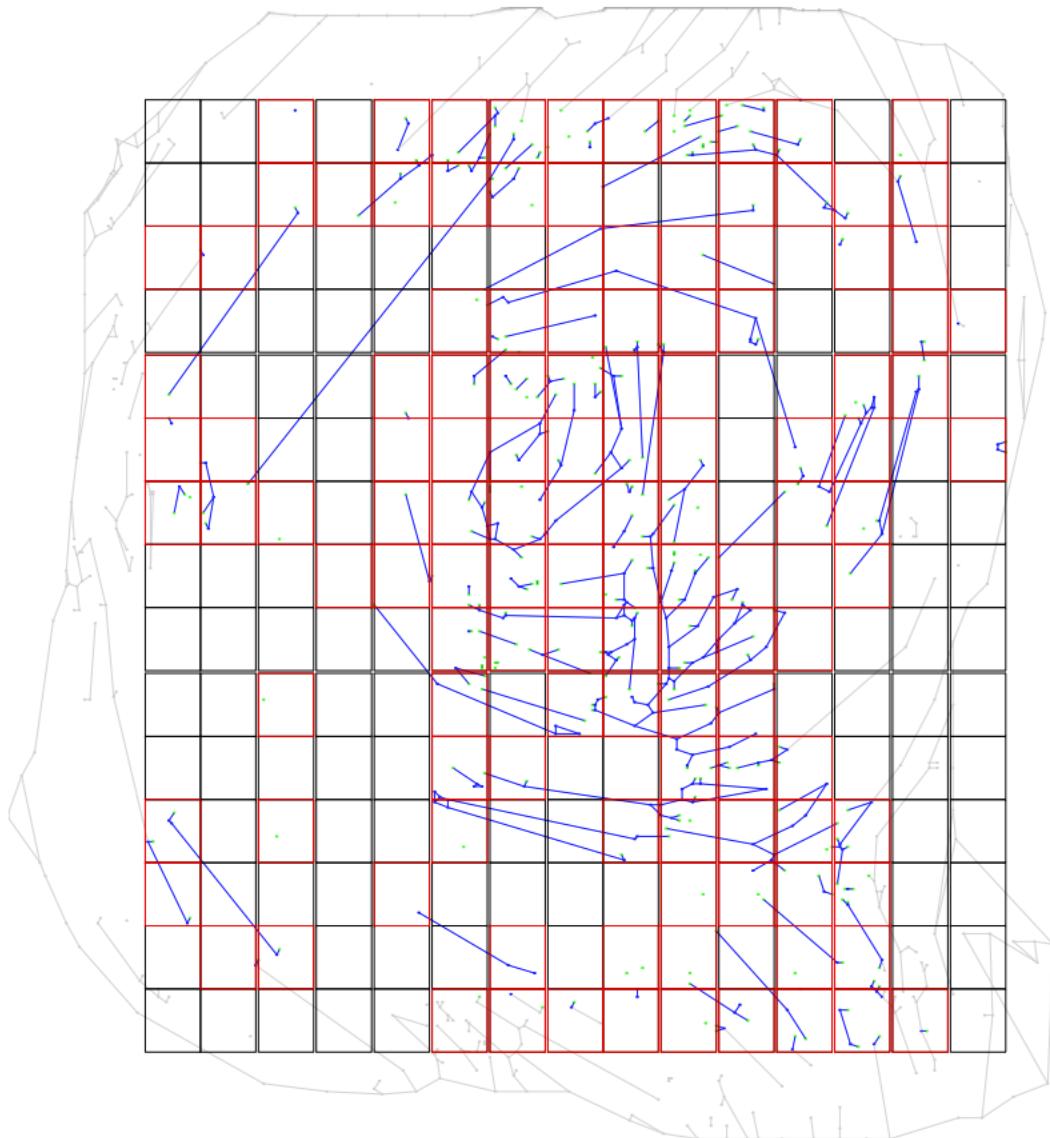


Figura 3.13: Rappresentazione della matrice sull’impronta 01_Ay_01_00.
Le celle rosse rappresentano celle con nodi al proprio interno, mentre le celle nere rappresentano celle totalmente vuote.

Capitolo 4

Risultati

In questo capitolo verranno approfonditi e illustrati nel dettaglio i metodi di scoring impiegati per l'analisi e la valutazione dei risultati ottenuti. Verrà data particolare attenzione ai criteri utilizzati per misurare l'efficacia e l'efficienza di ciascun algoritmo. Oltre a questo, verranno presentati i vari test eseguiti per ogni singolo algoritmo discusso durante lo sviluppo della ricerca. Ogni test sarà descritto in modo completo, includendo le condizioni di esecuzione, i parametri utilizzati, i risultati ottenuti e una discussione critica sulle prestazioni. L'obiettivo è fornire una visione chiara e dettagliata delle modalità con cui sono stati valutati i risultati, mettendo in evidenza i punti di forza e le aree di miglioramento per ogni approccio adottato.

4.1 Metodi di Scoring

Per poter fornire una valutazione accurata ai test eseguiti, è stato necessario sviluppare un metodo che permetesse di attribuire un punteggio il più realistico e affidabile possibile alle coppie di impronte digitali analizzate. A tale scopo, sono state ideate due strategie complementari.

La prima strategia, di natura più razionale e matematica, si basa principalmente sul rapporto tra i punteggi ottenuti dalle due impronte digitali, confrontando le loro caratteristiche in modo oggettivo. Questo approccio mira a quantificare le differenze o le somiglianze in modo sistematico e rigoroso, utilizzando parametri misurabili e precisi.

La seconda strategia, di carattere più empirico, si fonda invece su una particolare funzione progettata appositamente per attribuire il punteggio in base al grado di somiglianza percepito tra le impronte. In questo caso, il processo si affida a un'interpretazione più qualitativa, tenendo conto di fattori che potrebbero non essere pienamente descritti da un semplice confronto numerico, ma che contribuiscono comunque alla valutazione complessiva.

4.1.1 Scoring razionale

Il primo metodo di scoring che verrà analizzato si basa su una strategia puramente matematica e rigorosa. Questo approccio si concentra sulla determinazione del numero massimo di corrispondenze (o "match") che possono essere effettuate utilizzando un'impronta come riferimento. In altre parole, si prende una singola impronta e la si confronta con se stessa, cercando di calcolare la quantità massima di punti corrispondenti che possono essere individuati. Questo rappresenta il numero massimo teorico di corrispondenze possibili per quella specifica impronta.

Una volta ottenuta questa grandezza massima, si passa alla fase successiva del processo, che prevede il confronto tra l'impronta di riferimento e una seconda impronta da analizzare. A questo punto, si effettua il conteggio di quanti punti di corrispondenza si riescono a individuare tra le due impronte. Il risultato ottenuto da questo confronto viene quindi utilizzato per calcolare lo score. Lo score, in questo caso, è determinato dal rapporto tra il numero effettivo di corrispondenze trovate nel confronto con la seconda impronta e il numero massimo di corrispondenze possibili, il quale era stato precedentemente calcolato. Questa soluzione è applicabile a ogni algoritmo sviluppato in quanto è basato unicamente sul numero di corrispondenze riscontrate per ogni impronta.

L'equazione 4.1 mostra quindi la formula sviluppata.

$$\text{Score} = \frac{M_{\text{eff}}}{M_{\text{max}}} \cdot 100 \quad (4.1)$$

Dove:

- M_{eff} è il numero effettivo di match trovati confrontando l'impronta di riferimento con la seconda impronta.
- M_{max} è il numero massimo di match possibili, determinato confrontando l'impronta di riferimento con se stessa.
- Moltiplicando il risultato della divisione $\cdot 100$ si ottiene il valore percentuale.

4.1.2 Scoring empirico

Il secondo metodo analizzato si basa su un approccio empirico specificamente progettato per migliorare l'analisi delle corrispondenze nei nodi all'interno degli algoritmi di ricerca. Questo approccio si concentra sull'assegnazione di un peso basato sulla differenza tra i nodi dello stesso tipo, considerando la loro posizione in celle precise di una matrice.

In pratica, il metodo attribuisce uno score a ciascuna cella della matrice in base alla differenza osservata nel numero di nodi presenti. Più alta è la differenza nel numero di nodi tra celle simili, più basso sarà lo score assegnato a quella cella. Questo riflette l'idea che una maggiore discrepanza nel numero di nodi tra celle dovrebbe diminuire il peso dell'assegnazione, rendendo così meno significativo il valore di corrispondenza.

Per implementare questa logica, è stata creata una formula che simula questa situazione. La funzione è espressa come segue:

$$\text{Score} = 100 - f(\text{Diff}_{\text{bifurcation}}) - f(\text{Diff}_{\text{ending}}) \quad (4.2)$$

Dove $f(x)$ rappresenta la funzione creata appositamente per lo scopo e viene rappresentata come segue:

$$f(x) = \begin{cases} \frac{50}{\log(11)} \cdot \log(x+1) & \text{se } 0 < x \leq 10 \\ 50 & \text{se } x > 10 \\ 0 & \text{se } x = 0 \end{cases} \quad (4.3)$$

Dove x rappresenta la differenza nel numero di nodi tra celle simili. L'uso del logaritmo nella funzione è progettato per gestire le variazioni nella differenza dei nodi in modo non lineare, consentendo così una modulazione più fine dello score attribuito.

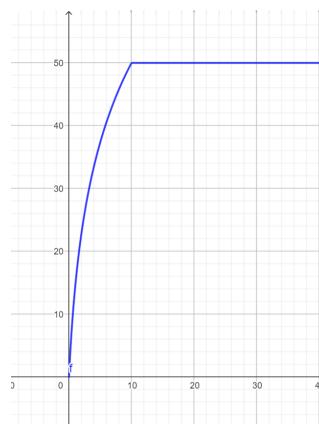


Figura 4.1: Grafico della funzione di scoring

4.2 Analisi dei test

Per verificare la somiglianza delle impronte digitali utilizzando i metodi descritti nel capitolo 3, sono stati condotti diversi test, ciascuno con parametri differenti. L'obiettivo è stato selezionare 6 gruppi dal dataset, includendo solo quelli con più di 30 acquisizioni di impronte, al fine di garantire coerenza nei test. Successivamente, sono state analizzate le impronte degli studenti e delle studentesse di riferimento di ciascun gruppo.

Per ogni dito di ciascun riferimento, sono state esaminate le impronte corrispondenti di ogni membro appartenente ai gruppi familiari presi in esame. Dopo aver analizzato tutte le dieci dita per ogni riferimento, è stata calcolata la media aritmetica dei risultati per ciascun gruppo familiare. Le tabelle che verranno presentate successivamente seguono questa struttura: per ogni riferimento, identificato dal suo gruppo di appartenenza, vengono riportate le percentuali di corrispondenza riscontrate per ogni gruppo familiare considerato. Infine, per ogni riferimento, viene evidenziato in grassetto la corrispondenza più alta trovata.

In totale, i test hanno coinvolto 18 persone per un totale di 2114 confronti.

4.2.1 Risultati algoritmo DFS

L'algoritmo DFS è stato testato in due modalità, ciascuna delle quali con due diversi set di parametri. Nel primo test, l'algoritmo è stato eseguito senza il preprocessing dei bordi, mentre nel secondo test è stato utilizzato dopo aver processato l'impronta dai bordi esterni. Entrambe le modalità sono state valutate con due configurazioni distinte di parametri.

Nel primo scenario, si è cercato di individuare sotto-grafi di cardinalità pari a 3, creati partendo da una box di 15 pixel e confrontando i risultati all'interno di una box di 30 pixel. Nel secondo scenario, invece, la ricerca ha riguardato sotto-grafi di cardinalità pari a 5, sempre generati da una box di 15 pixel e comparati in una box di 30 pixel.

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	39.2	32.9	32.4	35.9	34.5	21.0
02	34.9	27.1	29.0	28.4	32.5	22.3
06	41.1	35.4	39.5	38.4	37.4	26.7
08	35.9	31.7	32.6	40.4	39.4	23.3
13	36.8	34.2	34.5	38.4	41.7	23.3
18	36.5	33.5	30.4	35.1	34.2	16.7

Tabella 4.1: DFS senza preprocessing
cardinalità = 3, box creazione = 15, box comparazione = 30

CAPITOLO 4. RISULTATI

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	42.4	34.8	34.2	38.2	36.6	22.0
02	35.1	28.8	28.3	27.9	31.2	21.5
06	43.8	38.4	43.4	43.9	38.9	28.3
08	38.6	34.9	33.8	42.5	39.4	23.5
13	38.7	35.3	35.1	40.1	42.7	23.3
18	37.2	34.4	30.1	35.0	33.2	15.5

Tabella 4.2: DFS con preprocessing
cardinalità = 3, box creazione = 15, box comparazione = 30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	4.8	4.3	4.4	4.0	4.9	2.6
02	4.7	3.8	3.9	3.6	4.4	2.9
06	5.6	4.2	4.8	4.9	5.2	2.9
08	4.6	3.9	4.3	5.8	5.3	2.8
13	5.1	5.4	5.2	5.9	5.8	2.8
18	4.8	4.6	4.4	4.5	4.7	2.1

Tabella 4.3: DFS senza preprocessing
cardinalità = 5, box creazione = 15, box comparazione = 30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	9.9	9.9	8.4	11.1	9.2	5.4
02	10.0	8.2	8.1	7.8	9.3	6.1
06	12.6	11.6	11.8	10.9	9.7	8.1
08	9.6	10.1	9.8	14.6	10.7	6.7
13	10.1	10.5	10.1	11.9	12.3	5.7
18	10.3	10.6	9.0	11.0	9.0	4.2

Tabella 4.4: DFS con preprocessing
cardinalità = 5, box creazione = 15, box comparazione = 30

4.2.2 Risultati algoritmo BFS

L'algoritmo BFS è stato testato seguendo lo stesso approccio dell'algoritmo DFS. Sono state utilizzate due modalità di test con due set di parametri: una senza preprocessing dei bordi e una con preprocessing. I test hanno riguardato la ricerca di sotto-grafi di cardinalità 3 e 5, utilizzando box di 15 e 30 pixel.

CAPITOLO 4. RISULTATI

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	24.9	21.4	21.5	22.7	22.6	14.2
02	24.4	20.0	20.3	19.5	22.3	15.9
06	22.9	21.9	23.4	21.1	21.1	16.2
08	25.0	22.2	23.0	27.6	27.5	16.6
13	26.3	24.3	24.6	27.0	28.6	16.7
18	25.5	24.2	21.6	24.0	24.2	12.4

Tabella 4.5: BFS senza preprocessing
cardinalità = 3, box creazione = 15, box comparazione = 30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	41.6	34.6	33.3	37.5	36.7	21.9
02	35.0	28.8	28.4	27.9	31.2	21.7
06	43.8	37.8	44.4	44.9	38.4	28.2
08	38.9	35.7	34.0	42.0	39.5	23.9
13	38.8	35.7	34.7	40.0	42.7	23.2
18	36.6	35.0	29.8	34.5	33.2	15.3

Tabella 4.6: BFS con preprocessing
cardinalità = 3, box creazione = 15, box comparazione = 30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	4.8	4.3	4.4	4.0	4.9	2.6
02	4.7	3.8	3.9	3.6	4.4	2.9
06	5.6	4.2	4.8	4.9	5.2	2.9
08	4.6	3.9	4.3	5.8	5.3	2.8
13	5.1	5.4	5.2	5.9	5.8	2.8
18	4.8	4.6	4.4	4.5	4.7	2.1

Tabella 4.7: BFS senza preprocessing
cardinalità = 5, box creazione = 15, box comparazione = 30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	9.9	9.9	8.4	11.1	9.2	5.4
02	10.0	8.2	8.1	7.8	9.3	6.1
06	12.6	11.6	11.8	10.9	9.7	8.1
08	9.6	10.1	9.8	14.6	10.7	6.7
13	10.1	10.5	10.1	11.9	12.3	5.7
18	10.3	10.6	9.0	11.0	9.0	4.2

Tabella 4.8: BFS con preprocessing
cardinalità = 5, box creazione = 15, box comparazione = 30

4.2.3 Risultati algoritmo Singoli Nodi

Per l'analisi delle corrispondenze dei singoli nodi, sono stati effettuati quattro test sull'algoritmo. Ogni test ha comportato un incremento progressivo della dimensione della box utilizzata per la corrispondenza con i nodi dell'altra impronta, per poterne osservare la mutazione dei risultati. Le dimensioni della box testate sono state 1, 2, 4 e 8.

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	2.1	2.1	1.9	2.0	2.2	1.6
02	2.8	2.6	2.4	2.2	2.6	2.2
06	1.7	1.9	1.7	1.4	1.7	1.4
08	2.3	2.1	2.4	2.7	2.8	2.1
13	2.7	2.6	2.5	2.7	3.1	2.3
18	2.9	3.1	2.7	3.0	2.9	2.0

Tabella 4.9: Singoli Nodi con box comparazione = 1

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	5.7	5.5	5.3	5.3	6.0	4.3
02	7.7	6.8	6.5	6.0	7.1	6.0
06	5.1	5.0	5.0	4.3	5.0	3.9
08	6.3	5.6	6.6	7.4	7.5	5.5
13	7.2	6.9	6.9	7.6	8.4	6.0
18	8.3	8.0	7.3	8.2	8.0	5.9

Tabella 4.10: Singoli Nodi con box comparazione = 2

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	13.8	13.1	12.6	12.8	14.4	10.4
02	18.0	16.1	15.3	14.4	16.6	14.2
06	12.5	12.2	12.1	10.6	12.4	10.0
08	15.1	13.8	15.3	17.6	17.5	13.0
13	16.7	15.9	16.3	17.5	19.9	13.9
18	19.1	18.5	16.9	18.8	18.7	13.5

Tabella 4.11: Singoli Nodi con box comparazione = 4

CAPITOLO 4. RISULTATI

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	28.3	26.7	25.8	26.1	28.6	21.7
02	35.2	31.4	29.5	28.3	31.5	27.4
06	25.4	24.7	24.6	21.8	24.5	21.7
08	30.1	27.7	30.3	34.8	33.5	25.9
13	31.7	30.2	31.3	33.5	37.7	27.5
18	36.9	35.2	32.6	35.6	35.3	26.3

Tabella 4.12: Singoli Nodi con box comparazione = 8

4.2.4 Risultati algoritmo Matrice

L'algoritmo basato sull'idea della matrice è stato sottoposto a due modalità di test, ciascuna con due set di parametri distinti. Il primo tipo di test ha utilizzato l'approccio razionale per il calcolo dello score, come descritto nel capitolo 4.1.1. Il secondo test ha impiegato il calcolo dello score empirico, come illustrato nel capitolo 4.1.2. Ogni test è stato condotto utilizzando due dimensioni diverse della matrice: 15×15 e 30×30 .

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	47.0	41.0	38.4	40.0	38.1	27.0
02	39.4	34.3	32.8	30.8	33.0	26.1
06	48.5	44.3	49.5	46.5	41.7	33.3
08	41.6	38.7	36.7	43.5	38.7	27.3
13	41.5	38.8	37.5	39.2	42.8	26.7
18	40.0	38.0	33.9	37.3	34.8	20.1

Tabella 4.13: Score razionale, matrice 15×15

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	27.0	23.8	21.5	22.2	21.6	14.8
02	24.0	21.1	19.7	18.2	19.9	15.6
06	26.7	24.4	27.4	24.9	22.2	16.8
08	24.0	22.3	20.7	24.8	21.9	15.3
13	24.9	23.7	22.1	22.6	25.0	15.0
18	24.5	23.4	20.3	22.3	20.8	11.2

Tabella 4.14: Score razionale, matrice 30×30

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	28.0	34.2	33.4	30.4	36.6	30.6
02	39.0	30.3	34.1	32.4	40.3	40.0
06	40.8	39.1	42.2	40.2	39.8	42.3
08	47.1	50.4	44.5	41.0	35.5	43.0
13	50.1	51.1	43.4	47.7	45.0	50.9
18	47.9	49.4	49.5	52.1	47.2	49.3

 Tabella 4.15: Score empirico, matrice 15×15

	Gruppo 1	Gruppo 2	Gruppo 6	Gruppo 8	Gruppo 13	Gruppo 18
01	4.1	2.9	4.1	3.2	3.6	4.6
02	2.1	1.9	1.9	2.8	2.1	3.4
06	9.2	11.6	11.4	7.4	7.6	5.0
08	3.3	3.0	6.7	5.2	4.5	4.1
13	11.2	10.0	9.8	7.2	11.7	8.4
18	5.7	7.2	7.5	10.2	7.9	9.8

 Tabella 4.16: Score empirico, matrice 30×30

4.2.5 Considerazioni

Una volta esposte le tabelle riportanti i risultati forniti da ogni algoritmo, è doveroso trarre delle considerazioni in merito ai risultati ottenuti.

- **DFS** ha mostrato prestazioni solide, con un notevole miglioramento osservato grazie al preprocessing. Questo suggerisce che l'algoritmo è particolarmente efficace quando supportato da un'adeguata fase preparatoria.
- **BFS** ha ottenuto risultati molto simili a quelli di DFS, ma il miglioramento delle performance dovuto al preprocessing è stato meno marcato rispetto a DFS. Questo indica che, mentre BFS è competitivo, potrebbe trarre meno vantaggio dalle ottimizzazioni di preprocessing.
- **L'algoritmo di analisi dei nodi** ha mostrato risultati costanti in tutti i test, il che dimostra una buona affidabilità e coerenza nelle sue prestazioni.
- **L'algoritmo basato sulla Matrice** ha raggiunto eccellenti risultati quando valutato tramite lo scoring razionale. Tuttavia, i risultati ottenuti con il test empirico non sono stati altrettanto soddisfacenti, suggerendo che l'algoritmo potrebbe avere delle limitazioni quando applicato a scenari pratici.

Conclusione

In conclusione, nel corso di questa tesi è stata esplorata in dettaglio la storia e le caratteristiche delle impronte digitali, i metodi di classificazione storici e moderni, e il loro utilizzo nello scenario forense. È stata effettuata un'analisi approfondita del dataset, che ha permesso di sviluppare e testare gli algoritmi ideati, e i risultati ottenuti sono stati valutati criticamente per verificarne l'efficacia.

I risultati ottenuti attraverso lo sviluppo e l'applicazione degli algoritmi proposti hanno dimostrato una significativa efficacia nel confronto e nella correlazione delle impronte digitali con gruppi familiari specifici. Gli algoritmi sviluppati hanno fornito risposte positive e soddisfacenti, evidenziando la loro capacità di identificare affinità tra le impronte digitali analizzate e i gruppi di riferimento. Questo successo rappresenta un passo importante verso il perfezionamento delle tecniche di analisi forense e conferma la validità di questo approccio innovativo.

Tuttavia, il campo della biometria digitale è in continua evoluzione e offre ampie opportunità per ulteriori sviluppi. In futuro, sarebbe interessante espandere l'analisi per includere non solo le minuzie delle impronte digitali, ma anche la forma e la direzione delle creste. Un'analisi più dettagliata di questi aspetti potrebbe migliorare ulteriormente la precisione del confronto tra impronte digitali. Inoltre, l'implementazione di tecniche di preprocessing avanzate potrebbe contribuire a ridurre il numero di nodi superflui e affinare la qualità dei risultati, rendendo gli algoritmi più robusti ed efficienti.

Le prospettive future includono quindi l'ottimizzazione degli algoritmi esistenti e l'esplorazione di nuove metodologie che possano integrare una comprensione più approfondita delle caratteristiche delle impronte digitali. Questo approccio permetterebbe non solo di affinare le tecniche di identificazione forense, ma anche di ampliare le loro applicazioni pratiche in ambito investigativo e di sicurezza.

Bibliografia

- [1] Eric Aigbogun, Chinagorom Ibeachu, and Ann Lemuel. Fingerprint pattern similarity: a family-based study using novel classification. *Anatomy*, 13(2):116–121, 2019.
- [2] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
- [3] Brian E Dalrymple. Fingerprints. *The Forensic Laboratory Handbook: Procedures and Practice*, pages 117–141, 2006.
- [4] Davide Maltoni, Dario Maio, Anil K Jain, Salil Prabhakar, et al. *Handbook of fingerprint recognition*, volume 2. Springer, 2009.
- [5] Nalini Ratha and Ruud Bolle. *Automatic fingerprint recognition systems*. Springer Science & Business Media, 2007.
- [6] Robert Važan. Sourceafis algorithm, 2024.
- [7] Wikipedia. Giovanni Gasti — Wikipedia, the free encyclopedia, 2024.

Ringraziamenti

Giunti alla conclusione di questo percorso, è doveroso ringraziare infinitamente il Prof. Alessandro Dal Palù, nonché mio relatore, per la sua disponibilità nel chiarire ogni dubbio e nel supervisionare la ricerca svolta durante il periodo di tirocinio e scrittura di questa tesi.

Ringrazio inoltre tutti i professori e l'Università degli Studi di Parma per avermi fornito tutte le nozioni e le risorse per la mia formazione in questi anni accademici.

Un ringraziamento speciale va alla mia famiglia, la quale ha sempre rappresentato un pilastro fondamentale nella mia vita.

A mia madre Anna e mio padre Massimo, per il sostegno incondizionato che non mi hanno mai fatto mancare, per avermi cresciuto con sani principi e con l'amore profondo che solo dei genitori possono dare.

A mio fratello Luca, per tutto il supporto e per i consigli dati con la saggezza e la premura di un fratello maggiore, sempre pronto a guidarmi.

A Roberta, che mi ha sempre incoraggiato a dare il massimo e a non arrendermi di fronte agli ostacoli che ho incontrato lungo il cammino.

Ai miei nonni Iole e Silvio, che con il loro affetto e la loro costante presenza mi hanno sempre spinto a inseguire i miei sogni.

Desidero esprimere la mia profonda gratitudine ai miei cari amici e compagni di viaggio, Michele, Filippo e Simone. Senza di loro, questo percorso sarebbe stato infinitamente più arduo. Sono davvero fortunato ad aver incontrato delle persone meravigliose come loro, che hanno condiviso con me ogni passo di questa grande avventura.

Ringrazio infine i miei amici Otto, Davide, Samuele e tutti i compagni di corso, per aver alleggerito la quotidianità nelle giornate di lezioni con una risata e bellissimi momenti di svago.

Grazie di cuore a tutti,
Marco.

